Títol:	*Seges Mobile*
Volum:	*1*
Alumne:	*Canela Villagrasa, Marc*
Director:	*Capó Escrivà, Omar Isaac*
Empresa:	*Grupo Godó*
Ponent:	*Urpí Tubella, Antoni*
Departament:	*ESSI*
Data:	*12 de juny de 2013*

Dades del projecte

Títol del projecte: Seges Mobile

Nom de l'estudiant: Canela Villagrasa, Marc

Titulació: Enginyeria Tècnica en Informàtica de Sistemes
Centre: Facultat d'Informàtica de Barcelona (FIB)
Universitat: Universitat Politècnica de Catalunya (UPC) (BarcelonaTech)

Crèdits: 22'5

Director: Capó Escrivà, Omar Isaac

Empresa: Grupo Godó

Membres del tribunal (nom i firma)

Secretari (Ponent): Urpí Tubella, Antoni

President: Mayol Sarroca, Enrique

Vocal: Llanta Salleras, Estanislau

Qualificació

Qualificació numèrica:

Qualificació descriptiva:

Data:

Seges Mobile

Canela Villagrasa, Marc

Índex

Capítol 1

Introducció

En aquest capítol es vol presentar els motius pels quals es va escollir aquest projecte, fer una petita introducció al projecte, introduir l'entorn de treball en el que es van dur a terme totes les etapes i donar una visió global de l'entorn tecnològic i les implicacions que ha tingut.

1.1 Motivació

Durant els últims deu anys he estat treballant com autònom principalment desenvolupant aplicacions en *HTML*, *JavaScript* i *PHP* orientades a navegadors web d'escriptori. Aquest projecte em donava l'oportunitat de treballar en una aplicació HTML5 dissenyada específicament per a dispositius mòbils i que disposava de via lliure per experimentar les últimes novetats en el camp web. L'aplicació s'anava a desenvolupat en un context ideal per a fer el projecte de fi de carrera. Per aquest motiu vaig contactar amb el responsable de desenvolupament de Grupo Godó (Omar Capó) i li vaig proposar de presentar aquest projecte com a PFC.

Personal i professionalment aquest desenvolupament suposava una oportunitat de treballar en un departament més gran del que normalment treballo i això em podia aportar noves metodologies i el fet de conèixer noves tecnologies que normalment estan fora del meu àmbit de treball. Malgrat que la meva formació ha sigut l'enginyeria tècnica de sistemes informàtics, la carrera professional m'ha desviat cap a projectes més propis d'una formació de gestió.

Tot i que jo no entrava a treballar directament en l'equip de desenvolupament de Grupo Godó, el fet de que en el departament es treballés amb metodologies àgils, *Scrum*, i fessin servir tècniques com el *Pair Programming*, feia que tingues que adaptar-me a aquests cicles de desenvolupament. Per un altre costat, un punt interessant era el cicle d'integració continua que tenien, així com l'ús de les tecnologies més punteres en el costat del servidor.

El projecte també ha suposat un repte perquè s'havia de desenvolupar sota una sèrie de requeriments pressupostaris i d'integració amb una eina ja existent. Això ha permès el treball amb altres parts del departament i el disseny i desenvolupament conjunt dels serveis de comunicació entre les dos aplicacions.

1.2 Descripció del projecte

El projecte exposat en aquesta memòria consisteix en el disseny i implementació d'una versió reduïda i en format per a dispositius mòbils d'una aplicació web d'escriptori anomenada Seges que ha sigut desenvolupada pel departament de sistemes de Grupo Godó. Aquesta aplicació serveix per a fer el seguiment de les activitats comercials de l'empresa Publipress Media[1].

Seges permet als comercials enregistrar totes les seves activitats de manera que es puguin fer seguiments del que planifiquen i del que finalment s'ha executat. Una altra característica del sistema és que té una ordenació jeràrquica de tots els seus usuaris. D'aquesta manera es poden aplicar filtres a l'hora de mostrar les cites/tasques del sistema, permetent que els responsables de cada secció puguin veure i assignar tasques a ells mateixos i a tots els membres del seu equip (per sota en la jerarquia), i evitant que puguin consultar les que estan assignades a persones per sobre del seu nivell jeràrquic.

Finalment cal remarcar que les dades que gestiona l'aplicació estan recuperades directament des de l'ERP[2] de Grupo Godó a través dels serveis de Seges, assegurant així una integritat entre les dades.

1.3 Descripció de l'entorn empresarial

Grupo Godó és una empresa dedicada a la comunicació audiovisual la qual té la seva principal font d'ingressos en la venta d'espais publicitaris. Per aquest motiu el grup disposa de l'empresa Publipress Media que s'encarrega de realitzar totes les tasques de comercialització de tots els espais publicitaris repartits per tots els medis. Publipress Media gestiona la publicitat de La Vanguardia, El Mundo Deportivo, Rac 1, Rac 105, 8TV, les respectives webs, etc. Per a realitzar aquestes tasques disposa de diferents seus per tot el territori espanyol situant les principals a Barcelona i Madrid.

Publipress Media, per tal de poder portar un seguiment de totes les activitats comercials que realitzen els seus treballadors volien una aplicació àgil per tal de permetre als comercials informar d'una forma còmoda i ràpida dels resultats o modificacions de les planificacions. Un cop iniciat el desenvolupament de Seges es va veure que seria interessant una ampliació que aportés la possibilitat de realitzar totes aquestes tasques des de dispositius mòbils per permetre realitzar "sobre el terreny i en temps real" consultes, modificacions o informes sobre les cites planificades. Amb aquests requeriments neix **Seges Mobile.**

1 Empresa pertanyent a Grupo Godó
2 Enterprise Resource Planning (dins Grupo Godó s'utilitza un sistema SAP)

1.4 Entorn tecnològic

L'aplicació s'ha implementat sota la supervisió del departament de desenvolupament de Grupo Godó. Aquest departament imposa tot un seguit de normes que s'han de seguir en tots els projectes per tal d'evitar una dispersió metodològica i de tecnologia evitant així que els projectes externs s'implementin de qualsevol manera i compliquin el posterior enteniment o manteniment. Un dels motius més importants que han portat a aquest funcionament és el fet de que el departament subcontracta un volum important de les tasques a terceres empreses i es vol evitar un descontrol forçant a tots els proveïdors a seguir l'estàndard del departament.

Els projectes es gestionen seguint la metodologia *Scrum*, realitzant *sprints* de dues setmanes. Això implica que cada quinze dies s'hagi de fer la demostració de les tasques desenvolupades per tal de veure com evoluciona el producte, i en aquest punt, prendre les decisions per corregir les coses que no siguin adequades pel client. També es replanifica les tasques pel següent sprint.

El codi font de tots els projectes està allotjat en servidors de *subversion* dins les màquines de l'empresa. D'aquesta manera es força que el desenvolupament s'hagi de realitzar dins la xarxa privada de l'empresa bé de forma presencial o mitjançant xarxes privades virtuals amb el nivell de seguretat que això implica. Des de un punt de vista tècnic, l'estàndard de grupo Godó especifica que s'ha d'usar Java, i tot un conjunt de *frameworks* relacions com *Spring*, *JPA*, *ZK*, *CXF, etc.,* així com la manera d'usar-los. Per dur a terme la configuració del projecte i controlar el seu cicle de vida s'utilitza *Maven*. Tot el procés sempre està controlat per un cicle d'integració continuada que fa que cada dia es compili tot el codi i s'executin tots els testos i es mesurin les mètriques de qualitat amb eines com *Sonar*.

1.5 Organització de la memòria

Aquesta memòria està organitzada en vuit capítols que agrupen la introducció al projecte, especificació, el disseny i finalment les conclusions.

En el primer capítol es pretén donar una visió global de l'entorn on va sorgir el projecte, quines necessitats hi havia i com s'integra el projecte dins tota l'estructura. També s'expliquen les motivacions que van portar aquest projecte a convertir-se en el PFC i tot l'aprenentatge que ha suposat fer un projecte d'aquestes característiques.

En el segon i tercer capítol s'explica en detall quines són les necessitats del projecte i quines son les solucions finals que es van adoptar per implementar tot el sistema de la millor forma possible. En aquests capítols s'expliquen les controvèrsies entre les diferents possibles solucions i els possibles resultats de cada elecció.

En el quart i cinquè capítol s'explica en detall el disseny i la implementació de les parts més importants de l'aplicació, intentant destacar els punts més importants dins la lògica del sistema i els aspectes més interessants de la implementació.

En el sisè capítol es detalla el procés d'implantació i seguiment de l'aplicació i totes les proves que es van realitzar abans de donar el projecte per acabat. Aquest seguiment permet ontenir conclusions sobre la utilitat real del projecte i un balanç global sobre l'esforç que ha resultat al final tot el desenvolupament.

Finalment, en els capítols setè i vuitè es presenten les diferents conclusions a les que s'arriba després d'haver efectuat tot el projecte. Aquestes conclusions aglutinen dos aspectes diferents: per una banda tenim l'estudi econòmic que ens presenta els números reals dels costos del desenvolupament permeten concloure el cost econòmic total del projecte i les desviacions finals. Per l'altra tenim les conclusions personals sobre l'experiència viscuda durant tot el projecte.

Capítol 2

Especificació

En aquest capítol es descriu el funcionament principal de l'aplicació Seges i s'acoten els requeriments i especificació de la versió mòbil que és el que dona peu al PFC.

2.1 Seges

Seges és una aplicació web d'escriptori desenvolupada sota un conjunt de requeriments funcionals. Aquesta permet a cada comercial crear les seves cites convidant altres comercials, i afegir informació extra com clients, marques, medis, observacions, empreses, etc. Una de les característiques importants de Seges és el fet que la informació dels clients, anunciants, marques comercials, etc., està vinculada amb el sistema SAP. Això permet un correcte seguiment de tots els detalls d'una cita i elimina la possibilitat de que petites variacions en el nom (o durant la introducció de les dades) causin la duplicació d'informació dins el sistema. Per evitar la problemàtica de que no tothom està autoritzat a donar d'alta nous elements dins del sistema SAP, s'ha creat la possibilitat de generar entrades temporals dins d'una cita per agilitzar l'entrada d'informació o per no forçar una decisió errònia en cas de dubte. D'aquesta forma es permet al comercial que en arribar a l'oficina pugui entrar a la versió d'escriptori de Seges on li apareixeran unes notificacions indicant que s'han creat entrades noves i aquestes s'han de validar contra el sistema triant l'opció correcte, o en cas que sigui realment una nova entrada, iniciant tot el procés d'una alta nova dins el sistema SAP.

Una altre aspecte molt important és el filtre jeràrquic que permet la supervisió de les tasques únicament d'un mateix o de totes les persones per sota en l'estructura jeràrquica. Això permet definir grups i responsables de forma que aquests últims tenen accés a consultar i modificar totes les cites de les persones al seu càrrec. També permet definir cites on es poden incloure ells mateixos o totes les persones per sota en l'estructura jeràrquica. El sistema filtra de forma automàtica totes les possibilitats de manera que no és possible que algú inclogui en una cita ningú que estigui per sobre en l'arbre jeràrquic.

Una de les necessitats més importants era poder realitzar un bon seguiment de les reunions entre clients i comercials. Per facilitar aquesta tasca es va dissenyar una divisió informativa dins

d'una cita de forma que certs aspectes quedessin duplicats en dues pestanyes anomenades *Pre* i *Post*. El que permet aquest sistema és que mentre la cita estigui en el futur, totes les modificacions es faran sobre la pestanya de *Pre*. Un cop la cita ja hagi sigut iniciada automàticament s'activa la pestanya de *Post* i a partir d'aquest moment només és possible realitzar modificacions en la pestanya de *Post*. Això permet detalls com per exemple tenir una cita programada amb tres clients simultanis i que al final només dos s'hagin presentat. Quan s'obre la cita per modificar-la i eliminar el client que ha fallat, aquestes modificacions només afecten la pestanya de *Post*, deixant la pestanya de *Pre* intacte i permetent en un seguiment posterior detectar quines coses no han sortit com estaven programades.

Una altra possibilitat que ha sorgit és la de permetre la coordinació entre diferents equips a l'hora de programar les accions comercials sobre un client. Per exemple, si un equip de venta de publicitat en paper té programada una visita amb un client i l'equip de venta d'espai publicitari web també vol visitar aquest client, és possible coordinar-se i programar una visita conjunta.

Amb aquest sistema és possible per part del cap d'un grup la delegació de tasques comercials amb una simple assignació a una cita de la persona sobre la que es vol delegar/programar la visita. També permet assignar altres companys si el responsable creu convenient.

2.2 Especificació del problema

Seges sobrepassa les funcionalitats que s'han d'implementar en la versió mòbil. L'objectiu, doncs, de la versió mòbil és disposar d'una eina que permeti consultar, modificar, crear i eliminar noves cites d'una forma àgil, sense necessitat de disposar d'un ordenador de sobretaula o d'una pantalla amb prou resolució. Això permetrà una dinàmica de treball més àgil. També es demana que les possibilitats no quedin minvades en la mesura del possible en el moment que el dispositiu perdi la connexió amb la xarxa.

2.3 Requeriments funcionals

En aquest apartat s'expliquen els requeriments funcionals que ha de tenir l'aplicació. En gran mesura són requeriments referents a la interfície d'usuari ja que es tracta d'una aplicació que basa quasi tot el funcionament en resposta a les accions que fan els usuaris. Els requeriments per a la versió Seges Mobile queden definits en els apartats següents.

2.3.1 Model de la cita

2.3.1.1 Diagrama entitat – relació

En els requeriments de l'aplicació es va especificar que era necessari poder duplicar un conjunt concret de dades perquè es donava el cas que s'havia fet una previsió i el resultat de la reunió havia canviat. Per exemple un comercial programava una cita per vendre publicitat de la web Mundo Deportivo, però en el moment d'efectuar la reunió, el responsable amb el que havien quedat no podia assistir. Finalment la reunió es produïa amb una altra persona i el tema de la reunió podia acabar sent venta de publicitat en l'emissora de radio Rac1. Per aquest motiu es va demanar poder duplicar un conjunt de camps particular. Per resoldre aquesta demanda es va crear una entitat nova anomenada *DetallCita* que emmagatzema tots els camps que eren susceptibles de canviar entre la "pre cita" i la "post cita". El conjunt principal de dades associada a una cita es distribueix entre dues classes: *Cita* i *DetallCita*:

El conjunt de dades invariants d'una cita queden definides en la classe Cita. A continuació es descriuen els camps:

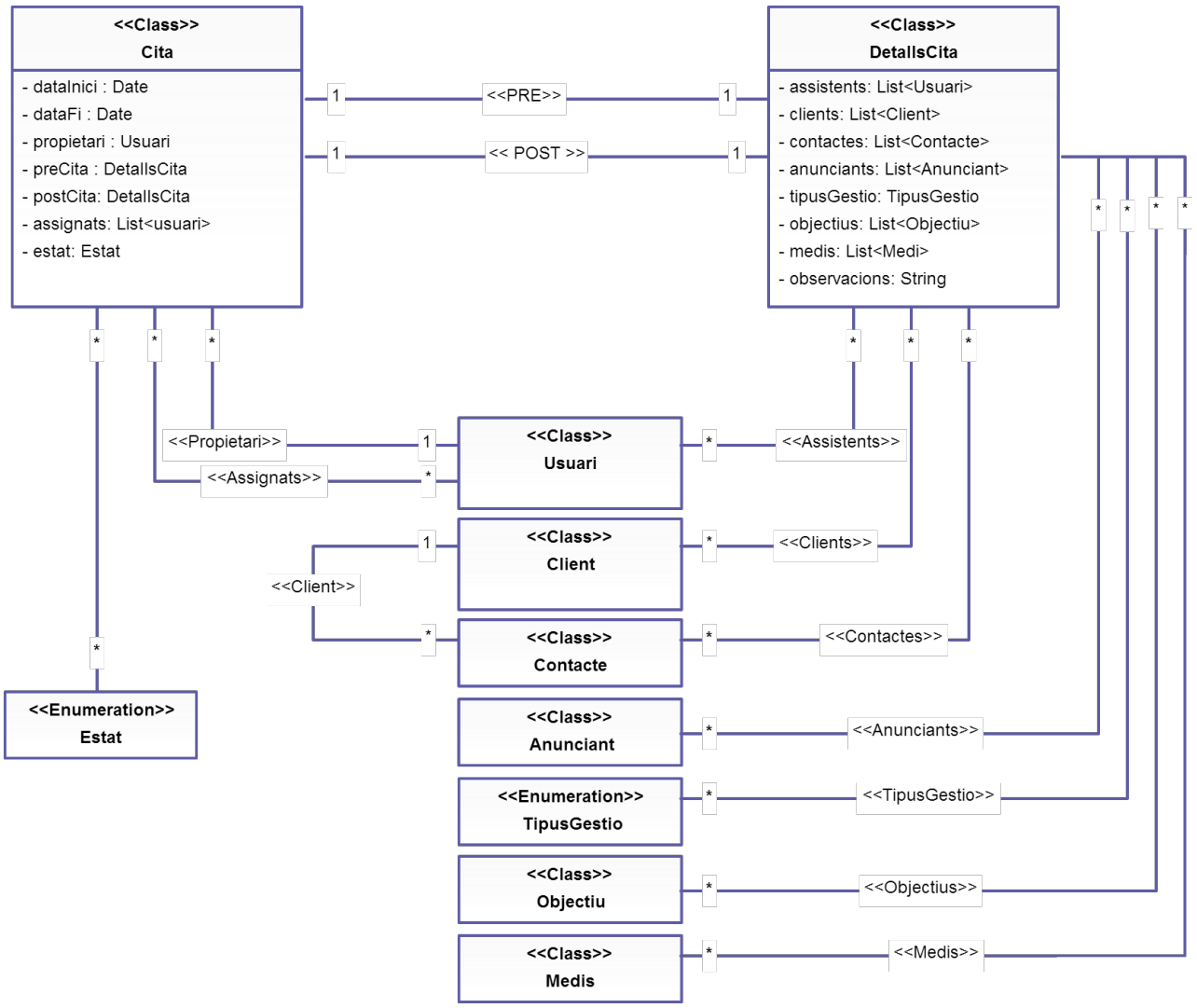
- Data Inici i Data Fi: són les dues dates en les que la cita està compresa.
- Estat: una cita pot tenir tres estats diferents: oberta, tancada i cancel·lada.
- Creador: és l'usuari que ha creat la cita.
- Assignats: és una llista dels usuaris que tenen assignada aquesta cita. Tots els usuaris assignats la podran visualitzar en el seu calendari i podran fer-ne un seguiment.
- Dades corresponents a la planificació de la cita (*Pre*): aquest camp està relacionat amb la classe *DetallCita* i permet emmagatzemar una instància que guarda la informació planificada per a la cita.
- Dades corresponents a la planificació de la cita (*Post*): aquest camp està relacionat amb la classe *DetallCita* i permet emmagatzemar una instància que guarda la informació resultant després d'una cita.

La segona classe definida és el *DetallCita*. Aquesta permet emmagatzemar múltiples instàncies dels següents camps:

- Assistents: són els usuaris que assistiran a la cita.
- Clients: són les empreses representades a la cita.
- Contactes: són les persones de contacte de l'empresa (client) del camp anterior i per tant són els subjectes que es personalitzaran el dia de la cita.
- Anunciant:
- Tipus de gestió: aquest camp permet indicar de quina manera es realitzarà la cita. Només pot ser d'un tipus.

- Objectius: en aquest camp s'especifica l'objectiu de la cita. Només pot haver-hi un.
- Mitjans: serveix per indicar quins són els mitjans propagandístics que es volen tractar en la cita.
- Observacions: permet introduir observacions o notes.

El següent diagrama del model de dades ens permet veure l'estructura que segueix una cita i com es relacionen les diferents classes que la defineixen.



2.3.1.2 Pre i Post

Alternació entre *Pre* i *Post*: quan es visualitzin els detalls o s'editin els camps d'una cita ha, de ser possible alternar entre els camps de la *pre* i de la *post*. Quan s'edita una cita només

s'ha de poder canviar les dades de la *pre* en cas que la cita no hagi començat o de la *post* un cop la cita hagi superat la data d'inici.

2.3.2 Visualitzar les cites

Visualització de cites: l'aplicació ha de permetre mostrar el volum de cites que s'estipuli en funció de la lògica funcional. En cas que totes les cites no es puguin visualitzar per la pantalla, la interfície ha de proporcionar un sistema de desplaçament vertical per permetre visualitzar totes les cites que queden fora del camp de visió.

Resum visual: s'ha de poder obtenir informació d'una cita en un primer cop d'ull sense la necessitat d'obrir-la per veure els detalls. Els camps preferits són: estat, clients i creador.

Cites filtrades: aquestes cites seran les assignades a l'usuari autenticat i les de tots els usuaris que estiguin per sota seu en l'escala jeràrquica.

La forma en com es mostren les cites ha de ser còmoda i en diferents ordenacions. Per complir aquest motiu es vol presentar les cites en tres formats de vista: cites diàries, cites mensuals i cites en mode llista.

2.3.2.1 Mode llista

Ha de permetre visualitzar les cites com si d'una llista es tractés. Ha de complir els següents requeriments:

Carregar més cites en el futur o en el passat: en aquest mode ha d'existir un sistema per poder carregar cites anteriors a la data on comença la llista o carregar més cites posteriors a la última cita.

2.3.2.2 Mode dia

Ha de permetre visualitzar totes les cites assignades a un dia. Ha de complir els següents requeriments:

Canviar de dia de forma còmoda: ha de permetre d'una manera directe canviar de dia per visualitzar les cites d'altres dates.

2.3.2.3 Mode mes

Ha de permetre visualitzar totes les cites assignades a un dia. També ha de mostrar un calendari on es mostri tot un mes. Aquest calendari ha de mostrar el número de cites que hi ha en cada dia. Ha de complir els següents requeriments:

Resum mensual directament en pantalla: aquesta vista ha de permetre d'una forma directe veure quantes cites hi ha programades per a tot el mes.

Navegació àgil: ha de permetre canviar de mes d'una forma ràpida.

2.3.3 Detalls d'una cita

Accés als detalls d'una cita: des de qualsevol de les tres vistes ha de ser possible accedir als detalls d'una cita d'una forma clara i senzilla.

Visualització còmoda dels detalls: ha d'existir un mètode pràctic que permeti visualitzar tots els detalls d'una cita encara que no hi càpiguen tots per pantalla. Per exemple, si surten ordenats un sota de l'altre, hauria de ser possible efectuar un desplaçament vertical per visualitzar els camps que queden fora de la pantalla.

Camps específics: es vol que una cita tingui els següents camps: data inici, data fi, propietari, estat i assignats.

Camps distribuïts: es vol que els següents camps estiguin duplicats i ordenats en dos agrupacions: Pre i Post. El motiu d'aquesta separació és que aquests camps poden canviar en el moment d'efectuar la reunió. Amb aquesta solució es pretén poder comparar la previsió inicial i la final sobre els següents camps: assistents, clients, contactes, anunciants, tipus de gestió, objectius, mitjans i observacions.

2.3.4 Editar una cita

Editar una cita: ha de ser possible editar tots els detalls d'una cita.

Salvar canvis: quan s'està en el mode edició d'una cita o editant qualsevol dels camps de la cita que tenen una vista pròpia, ha d'existir un botó que permeti guardar els canvis realitzats.

Descartar canvis: quan s'està en el mode edició d'una cita o editant qualsevol dels camps de la cita que tenen una vista pròpia, ha d'existir un botó que permeti descartar els canvis realitzats .

Data inicial i final: les dates d'una cita han de ser concordants. No s'ha de poder deixar la cita configurada amb la data de finalització anterior a la data d'inici. Quan s'edita una data i

aquesta superposa l'altra, automàticament l'altra s'ha de situar una hora per davant o per darrera en funció de la data que s'estigui editant.

2.3.4.1 Auto suggeriment

Auto suggeriment: els camps assignats, assistents, clients, contactes, anunciants, objectius i mitjans tenen una interfície que permet que a mesura que es va escrivint text es mostrin possibles suggeriments. Aquest suggeriments han de ser recuperats del servidor.

Llista de suggeriments: en el moment d'omplir un dels camps anteriorment citats ha de ser possible triar entre els elements que es suggereixen. Aquestes dades s'han de recuperar en temps real des del servidor el qual consulta les dades directament als servidors mestres de SAP de Grupo Godó.

Camp contactes filtrat: quan s'estan afegint contactes a una cita, els elements que es recuperen del servidor han d'estar filtrats en funció dels clients seleccionats. Si no hi ha cap client, es mostren tots els contactes. Si hi ha com a mínim un client, només es mostren els contactes d'aquell o aquells clients afegits.

Permetre nous elements: els camps clients, contactes i anunciant han de permetre la introducció d'un text diferent a qualsevol dels suggeriments proporcionats. Això es du a terme permetent la creació d'entitats temporals. Aquestes entitats temporals seran, a posteriori des de la interfície d'escriptori, vinculades (substituïdes) amb una entitat real del sistema o assignades a un nou procés d'alta en el sistema en cas necessari.

2.3.5 Crear una cita

Creació d'una cita: quan no s'està visualitzant o editant una cita ha de ser possible crear-ne una de nova.

2.3.6 Eliminar una cita

Eliminació d'una cita: s'ha de poder eliminar una cita de forma còmoda però intentant que no es pugui realitzar l'acció de forma accidental.

2.4 Requeriments no funcionals

Els requeriments no funcionals de l'aplicació es presenten agrupats en diferents apartats per descriure d'una forma més còmoda el seu propòsit.

2.4.1 Interfície d'usuari i factors humans

Controls tàctils: els controls de la interfície han de respondre al tacte de la pantalla per permetre una còmoda interacció entre l'usuari i l'aplicació.

Avís de canvis no guardats: quan s'està editant qualsevol element, s'han realitzat canvis i es vol sortir sense salvar ha de sortir un avís indicant que els canvis es perdran.

Avís en cas d'abandonar l'aplicació: al ser una aplicació web és molt habitual utilitzar les fletxes d'historial per anar enrere en la navegació. L'aplicació ha de mostrar un avís en cas que es detecti que l'usuari fa us d'aquestes fletxes.

2.4.2 Sincronització amb el servidor

Mantenir la sincronització: l'aplicació ha de mantenir una sincronització amb el servidor central sempre que hi hagi cobertura. Aquesta sincronització s'ha de realitzar en intervals definibles a partir de les proves de funcionament. És desitjable que la sincronització no es realitzi en intervals més grans de cinc minuts com a màxim.

2.4.3 Independència de l'estat de la connexió del dispositiu

Mode sense connexió: s'ha de permetre que l'aplicació continuï funcionant si s'ha perdut la connexió amb el servidor.

Dades locals: s'ha de mantenir un registre local de totes les cites per agilitzar-ne les consultes i no dependre sempre del servidor central.

Optimització del tràfic de dades: disminuir el tràfic amb el servidor gràcies a la memòria cau del sistema.

2.4.4 Seguretat

En quant a seguretat, l'aplicació ha d'implementar suport per al mètode corporatiu d'autentificació i ha d'implementar la funcionalitat de "*remember me*". Amb aquest suport es permet entrar a l'aplicació utilitzant les credencials globals dins de Grupo Godó i això vol dir que es poden utilitzar el mateix usuari i contrasenya que en la resta d'aplicacions corporatives. Queda, doncs, definit el requeriment funcional de la següent manera:

Connexió i desconnexió del sistema: ha de ser possible validar-se amb un usuari i tancar sessió per poder-se validar amb un altra usuari en cas que es vulgui. L'usuari i la contrasenya han de ser les mateixes que per la resta d'aplicacions corporatives.

2.4.5 Consideracions de hardware

L'aplicació hauria de funcionar sobre tots els dispositius mòbils del mercat. És imprescindible que funcioni sobre Android i concretament la versió que incorporen els Samsung Galaxy II ja que és el mòbil que l'empresa distribuirà entre els comercials. També és preferible que hi hagi compatibilitat amb dispositius iOS i Windows Mobile.

Capítol 3

Estat de l'art

En aquest capítol es presenten les alternatives que van anar sorgint durant l'anàlisi de les especificacions i s'analitzen els pros i contres atenent les especificacions del projecte.

3.1 Compatibilitat entre dispositius mòbils

Aquest ha sigut un dels factors més crítics a l'hora de decidir la tecnologia amb la que es programaria l'aplicació. Inicialment es demanava que l'eina funcionés amb Android ja que l'empresa va arribar a un acord amb la marca Samsung per obtenir el dispositiu Samsung Galaxy II per a tots els comercials de Publipress Media. Era obvi que el més recomanable era que l'aplicació funcionés en tots els dispositius, com a mínim amb *iOS* i Android, que representen la quota de mercat majoritària.

Avui en dia la majoria d'usuaris disposen de telèfons intel·ligents d'empresa o bé particulars els quals es divideixen entre *iOS* i Android d'una forma bastant equilibrada. Era per tant molt mala idea obligar els comercials a utilitzar el telèfon d'empresa per poder treballar amb l'aplicació ja que molts cops treballen fora de l'horari comercial i no sempre tenen el telèfon d'empresa a mà.

Així doncs quedava clar que l'aplicació havia de ser multi plataforma. Davant aquest dilema es van proposar les següents solucions:

- Fer dues aplicacions natives, una per a cada sistema mòbil.
- Utilitzar Adobe Air (reproductor Flash natiu) i dissenyar l'aplicació per a aquest sistema.
- Dissenyar l'aplicació com una aplicació web i preparar una aplicació contenidor amb un navegador incorporat que permetés el funcionament com si d'una aplicació nativa es tractés (Un contenidor per a cada sistema). Això presentava certs avantatges, com per exemple l'execució en segon pla i altres possibilitats d'interacció amb el sistema.
- Dissenyar l'aplicació com una aplicació web i fer-la funcionar en els navegadors natius de cada sistema.

La primera proposta, la qual presentaria els millors resultats, es va descartar per la implicació en costos que suposava la n-ària programació d'una aplicació en múltiples llenguatges per cobrir les plataformes més importants del mercat.

La segona també es va descartar perquè no és interessant dependre d'una tecnologia que no te gaire clar el seu futur i sobre la qual Apple s'ha pronunciat bastant en contra.

La discussió quedava doncs entre la tercera i la quarta proposta. La única diferencia entre les dues residia en si era rentable el cost extra dels contenidors respecte els avantatges que proporcionava. Aquest contenidor, en el cas que es va estudiar, corresponia a la tecnologia presentada pel producte comercial *PhoneGap*. L'únic avantatge real que es va trobar va ser el de permetre que l'aplicació seguís funcionant en segon pla. En la pràctica es va decidir que no valia la pena fer aquesta adquisició per estalviar un parell d'accions com a màxim per poder accedir a l'aplicació. Cal remarcar que tant *iOS* com Android permeten crear ens els respectius escriptori accessos directes a pàgines web i que fins i tot (en *iOS*) es pot configurar perquè s'executi a pantalla completa simulant una aplicació nativa.

Les tres primeres opcions també van ser descartades pel tema de la distribució específicament en cada un dels mercats de cada sistema i pel motiu que des del departament es segueix la política de no introduir més tecnologies de les necessàries per evitar un sobre esforç de recursos per estar al dia amb cada una de les tecnologies i mantenir-les.

Així doncs la decisió final va ser la implementació de l'aplicació utilitzant l'estàndard web. Aquesta decisió també ha permès que l'aplicació sigui compatible amb qualsevol plataforma que tingui un navegador compatible amb la última versió de *l'HTML*: *Android*, *iOS*, *Windows Mobile*, Nokia, etc. sense la necessitat de fer més inversió per poder fer l'aplicació compatible entre més sistemes el dia de demà.

3.2 Tecnologia web

Un cop encaminada la forma en com s'implementaria l'aplicació es va fer un estudi de les diferents eines que existien i que podrien ser d'utilitat a l'hora de desenvolupar l'aplicació. La llista de les opcions examinades va ser la següent:

- Dhtmlx Mobile Scheduler
- Sencha Touch
- Jquery Mobile

La primera opció experimentada va ser l'entorn de treball *Dhtmlx Touch Library*, el qual tenia una eina ja programada que s'assemblava molt a les necessitats inicials del projecte. *Dhtml Mobile Scheduler* és el nom que rep el mòdul fet utilitzant aquesta llibreria i que presenta un calendari amb tres vistes inicials (llista, dia i mes) i diferents opcions d'edició per als esdeveniments que s'hi podien guardar. Els problemes que van sorgir amb aquesta proposta van ser dos; per una banda el problema que es repetiria en totes les demés propostes estudiades, i és

la robustesa de l'aplicació quan funciona en dispositius mòbils. El principal defecte era que quan la pantalla es redimensiona bé sigui per un canvi de posició del dispositiu (invertint la vertical i la horitzontal) o per l'aparició del teclat (en Android el teclat causa que es redimensioni l'espai visible a diferència de *l'iOS* que mostra el teclat superposat) no s'adaptava bé a les noves mides i començava a efectuar comportaments imprevistos i fora del que s'esperava. L'altra gran problema va vindre per la poca extensibilitat de l'API del mòdul. L'API que presentava era molt limitada i posava moltes barreres al que es volia poder efectuar sobre una cita. Per aquests motius es va descartar el seu us.

La segona opció que es va estudiar va ser la utilització de l'entorn de treball per a mòbil *Sencha Touch*. Dins la seva web es va poder provar les demostracions i des de l'escriptori tot semblava una meravella. Era una eina bastant completa i tenia molt bona aparença. Els problemes van començar a sorgir quan es van començar les proves des del dispositiu mòbil. El que inicialment eren uns moviments ràpids i fluïts de sobte es van convertir en no tant fluids i un pel lents. Suposem que per algun tema d'optimització o potser d'alguna incompatibilitat amb el navegador natiu de *l'Android* el sistema funcionava més feixuc. El problema real va venir durant les proves de començar a girar la pantalla, canvis de mida, aparicions i desaparicions del teclat, etc. Durant aquestes proves l'eina no es va mostrar suficientment robusta i es va decidir descartar-la per evitar posteriors problemes d'usabilitat amb els usuaris finals, els quals no són tan hàbils en aquests temes i de segur que començarien a sorgir queixes sobre el sistema. La versió actual sembla que ha corregit bastants problemes i ara mostra un comportament més robust. És una bona opció a estudiar en futures aplicacions d'aquest tipus.

La tercera opció que es va estudiar va ser la branca orientada a mòbil de la famosa llibreria *jQuery*. Es van estar fent proves en la web i tot semblava funcionar prou bé, però un cop més, en el moment de començar les proves en els dispositius mòbils van començar a sortir els problemes d'incompatibilitats i comportaments estrany en el moment d'efectuar rotacions o entrar i sortir del teclat. Per altra banda, la quantitat d'artefactes disponibles era molt bàsica i realment no ens solucionava els aspectes principals que teníem en ment.

3.3 Protocols de transferència

En aquest punt del desenvolupament es va plantejar quines possibles opcions teníem per poder realitzar la **comunicació** entre el dispositiu i el servidor. Les opcions no eren gaires perquè s'estava limitat per les possibilitats del navegador el qual només brindava les següents:

- Rest / RESTFul
- Websocket
- HttpRequest (XHR)

La solució Rest va quedar descartada de bon començament perquè suposava haver de recarregar la pàgina cada vegada que es volia fer una comunicació amb el servidor. Això suposava el greu problema d'un sobre cost de l'ample de banda i el subseqüent temps de processat i presentació de l'aplicació, cosa que també consumeix bateria.

Websocket és una solució que ha aparegut amb la nova especificació de *l'HTML5*. Representa el funcionament clàssic entre client i servidor, permetent obrir una connexió persistent que dura tota l'estona que el client vol estar connectat. Aquest sistema te la clara avantatge que quan el servidor vol notificar qualsevol cosa pot fer-ho en temps real i no ha d'esperar que sigui el client qui pregunti si hi ha cap novetat. El problema d'aquesta solució resideix en que el servidor ha de mantenir el control de totes les sessions concurrents i en determinats sistemes això pot generar un sobrecost de recursos poc interessant. Un altra dels problemes és que en les connexions mòbils, les connexions i desconnexions a la xarxa es realitzen molt sovint i això causaria moltes finalitzacions de sessió inesperades amb els possibles problemes de pèrdua d'informació o generació extra de connexions al servidor deixant-ne moltes d'altres orfes i en espera de ser descartades per inactivitat per part del servidor. Un altra dels problemes que existien és el suport d'aquesta nova especificació, el qual no és total encara en tots els navegadors i podia generar problemes d'incompatibilitats. Per altra banda, per la naturalesa de l'aplicació que es volia implementar no era necessari mantenir connexions persistents, per tant es va descartar.

Finalment la utilització de connexions Rest asíncrones amb el servidor (XHR o AJAX) va ser la solució escollida.

3.4 Format de representació de dades

Un cop establert el sistema de comunicació amb el servidor va ser necessari determinar quin protocol de representació de dades es faria servir. Les opcions que es van posar sobre la taula van ser les següents:

- xml
- csv
- ics
- pla
- json

La transmissió de la informació en format *xml* generava un sobrecost en etiquetatge que podia ser letal sobre uns canals de comunicació de per si lents i amb un consum de dades limitat

com podien ser les connexions 3G. Per tant aquest protocol es va descartar en detriment d'un més compacte.

El format *csv* era massa bàsic per a la representació de les dades que volíem transmetre i per tant també va quedar descartat.

Ics és un format estàndard de calendari que es va fer famós a partir de l'aplicació *iCalendar* que acompanyava els productes d'Apple. També es coneix aquest format com *iCal* a causa del programa mencionat. El format permet la transmissió de calendaris i altres detalls. Això, en quant a la transferència del calendari amb les cites era una possible solució, però no permetia la transferència d'informació de tipus llistes per poder, per exemple, assistir els camps amb auto completat. Per aquest motiu es va decidir buscar un protocol que fos més flexible i permetés més varietat en el format.

Es va fer un petit estudi de com seria un possible protocol molt bàsic que s'ajustés al que necessitàvem. Després d'unes petites proves de format es va veure que realment no era possible transmetre la informació en un protocol pla ja que acabava sent necessari un conjunt de marques per poder estructurar la informació, per tant es va descartar i es va buscar un protocol ja existent.

Finalment i sense gaires dubtes, *JSON* va ser el protocol que més s'ajustava a les nostres necessitats. El conjunt d'avantatges era molt gran i en contra tenia més aviat poques coses. Era un protocol amb poca càrrega de marques, permetia una ràpida transformació des de *JavaScript* i permetia encapsular qualsevol tipus d'informació. En la part del servidor, *Spring* (entorn de treball basat en Java) incorpora molta compatibilitat amb aquest protocol i permet molta agilitat a l'hora de processar les peticions entrants sota *json*. Així doncs aquest va ser finalment el protocol seleccionat.

Capítol 4

Disseny

En aquest capítol es presenta el disseny de l'aplicació mòbil que es vol implementar. Degut a que una aplicació mòbil té una part molt important d'usabilitat, s'ha dividit el disseny entre l'aspecte visual i l'aspecte funcional. La intenció és justificar totes les decisions de disseny indicant el perquè s'ha pres cada decisió.

4.1 Decisions de disseny

Després d'un estudi dels requeriments i el posterior anàlisi, es van prendre una sèries de decisions que condicionarien el disseny i permetrien complir amb tots els requeriments de la millor forma possible. Aquestes decisions s'expliquen a continuació amb detall i justificació del perquè de cada elecció.

4.1.1 Elecció de la tecnologia

En aquest apartat s'ha decidit implementar l'aplicació utilitzant la tecnologia web HTML5 (i *JavaScript*). Una aplicació realitzada en HTML pot funcionar en qualsevol sistema que tingui un intèrpret web (navegador) que sigui compatible amb les últimes novetats de l'estàndard HTML. Això permet que qualsevol sistema operatiu del mercat pugui fer funcionar l'aplicació sempre que es compleixi l'estàndard.

Un punt interessant per utilitzar aquesta tecnologia són les novetats de l'última revisió de l'estàndard; la versió 5. Entre aquestes novetats apareixen la possibilitat d'utilitzar una memòria local del navegador per emmagatzemar arxius o dades amb una capacitat màxima de cinc megabytes (per defecte). També incorpora sistemes per guardar arxius en memòria cau de forma que una aplicació pugui arrancar i funcionar sense la necessitat d'estar connectat a la xarxa.

4.1.2 Transmissió de la informació

Per a transmetre la informació, la tecnologia finalment seleccionada ha sigut la comunicació asíncrona amb XHR (A*jax*). Amb aquest sistema es poden fer crides asíncrones en qualsevol moment i es permet una versatilitat molt gran en el tipus d'informació que s'envia o es rep des del servidor ja que utilitza l'estàndard HTTP per realitzar la transmissió d'informació.

4.1.3 Representació de dades

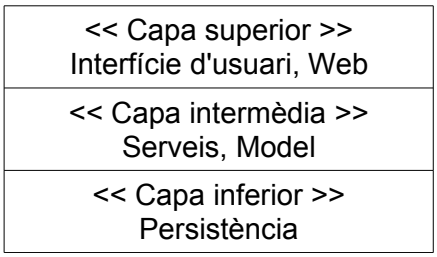
El format que finalment s'ha seleccionat per realitzar la codificació de les dades ha sigut l'esmentat *JSON*[3]. Bàsicament és el text que s'obté del procés de passar d'un objecte *JavaScript* a text.

Aquesta representació té l'avantatge que és molt lleugera i està suportada de forma nativa per les actuals implementacions de *JavaScript* de tots els navegadors. Un altra avantatge és que es pot interpretar directament una representació en aquest format utilitzant la funció *eval()* que incorpora *JavaScript*. Això permet una gran versatilitat a l'hora de tractar informació representada en aquest format.

Un altra punt a favor d'aquesta representació és que gràcies a la seva popularitat cada cop més llenguatges de programació incorporen eines per tractar amb aquest format. En el nostre cas Java presenta una gran compatibilitat i resulta molt fàcil recuperar i interpretar la informació que arriba al servidor en aquest format.

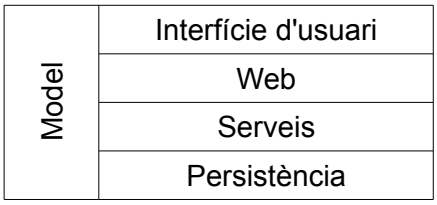
4.2 Arquitectura de l'aplicació

Per concordança amb la metodologia del departament es pretén dissenyar l'aplicació seguint el model de capes que es representa en el següent diagrama:



3 JSON: acrònim de JavaScript Object Notation. És un format per l'intercanvi de dades i està composat d'un subconjunt de la notació literal d'objectes de JavaScript

Aquesta arquitectura també es pot trobar molts cops distribuïda en quatre capes i una transversal:

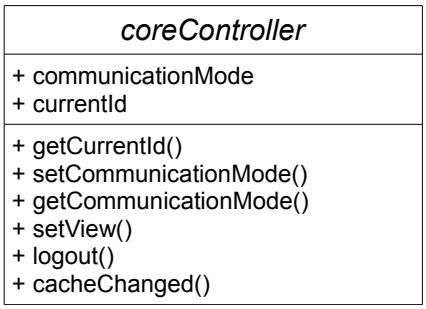


Aquest model permet efectuar modificacions en una capa sense que la resta de capes de l'aplicació es vegin afectades. També permet una distribució de la lògica segons les responsabilitats de cadascuna de les capes. Així doncs a la capa de persistència és on hi ha tota la lògia de guardar i recuperar la informació. A la capa de servei hi trobem tota la lògica pròpia del negoci (capa coneguda també com Business Layer o capa de negoci). Per últim a la capa de presentació tenim tota la lògica referent a la representació de la informació. Tal com es veu en l'últim diagrama, a vegades es subdivideix en dos capes; una pròpiament la representació i l'altra el controlador de les vistes. El model és transversal a totes les capes. A continuació es detallen totes les capes.

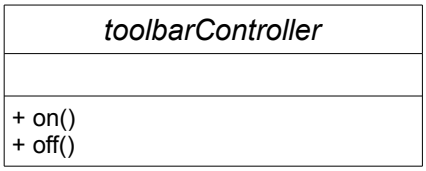
4.2.1 Lògica principal de l'aplicació

En aquest apartat s'agrupen els controladors encarregats de gestionar la capa de servei principal de l'aplicació. A continuació es detalla les funcions principals de cada controlador i el seu diagrama UML:

- coreController: aquesta classe s'ha creat amb la finalitat de realitzar el paper de nexe d'unió de tota la lògica. Disposa d'un conjunt de mètodes que permeten recuperar informació sobre diversos estats com per exemple l'estat de la comunicació amb el servidor, l'identificador de la cita actualment detallada, etc.



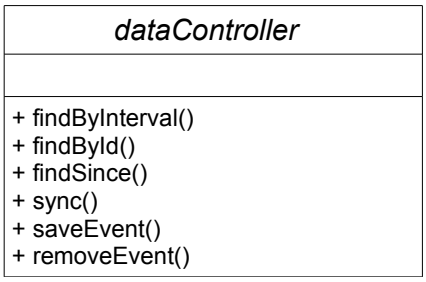
- toolbarController: aquesta classe té el propòsit de gestionar la lògica de les diferents barres que apareixen en la part superior o inferior en les diferents vistes. S'ha creat una classe específica per aquest menester per poder agrupar tota la lògica referent al funcionament de les barres superior i inferior.



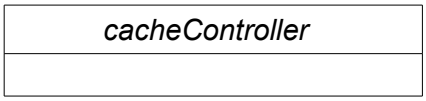
4.2.2 Lògica de l'indexat de cites i memòria cau

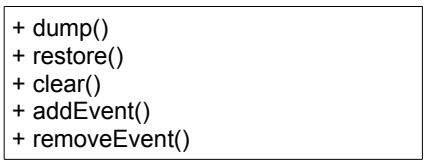
En aquest apartat s'agrupen els controladors destinats a gestionar tota la capa de persistència referent a l'emmagatzemament i gestió interna de les cites. Els controladors encarregats d'aquestes funcions són:

- dataController: aquesta classe s'ha creat amb la finalitat de rebre totes les peticions sobre el conjunt de cites. Aquest controlador interactua directament amb el controlador de la memòria cau i determina si cal comunicar-se amb el servidor per recuperar més cites. En cas que la memòria cau no tingui un interval suficientment gran per cobrir la demanda de cites aquest controlador s'encarregarà de gestionar les peticions amb el servidor i actualitzar la memòria cau amb la informació de les respostes. També s'ha creat amb l'objectiu de controlar el cicle automàtic de sincronització amb el servidor.



- cacheController: aquesta classe es crea amb l'objectiu d'emmagatzemar totes les cites i indexar-les de la forma més òptima per agilitzar l'accés. També cobreix la funció de gestionar la interacció entre la memòria cau del navegador i la memòria cau de l'aplicació mantenint una persistència de totes les cites de forma local.





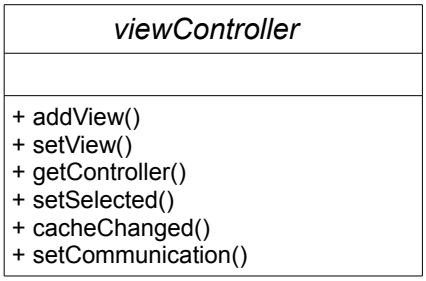
4.2.3 Lògica de presentació

En aquest apartat s'agrupen tots els controladors que s'encarreguen dels aspectes visuals de l'aplicació. Els controladors d'aquest grup es poden dividir en els diferents tipus següents:

- Controlador de les vistes
- Classe base d'on hereten tots els controladors visuals
- Controladors de les diferents vistes
- Controladors auxiliars amb tasques específiques

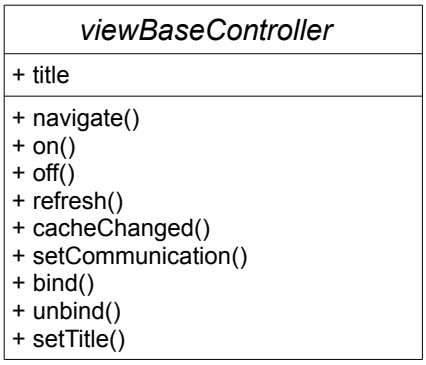
4.2.3.1 Controladors de les vistes

El controlador encarregat de gestionar les vistes és el *viewController*. Aquest controlador es crea amb l'objectiu de guardar un registre de totes les vistes i encarregar-se de replicar per tot el registre els diferents esdeveniments especificats en el disseny. El seu diagrama és el següent:



4.2.3.2 Classe base d'on hereten tots els controladors visuals

El controlador base des de on hereten tots els controladors de les vistes rep el nom de *viewBaseController.* S'ha creat amb l'objectiu de definir els mètodes base que han de tenir tots els controladors de les vistes. Entre les funcions base s'hi defineixen les que permeten implementar les accions de les barres, les que permeten activar i desactivar la vista, les referents a canvis en la memòria cau i les que permeten assignar funcions als esdeveniments de l'usuari. El seu diagrama és el següent:

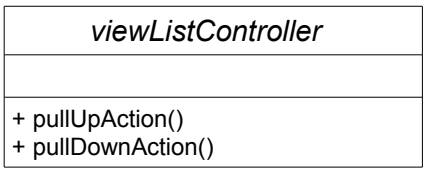


La resta de controladors que hereten d'aquest implementen també aquests mètodes i atributs.

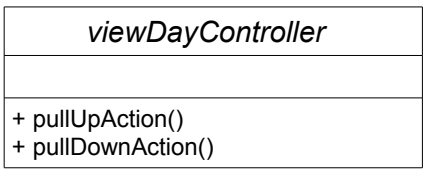
4.2.3.3 Controladors de les diferents vistes

Els controladors de les vistes s'han creat amb l'objectiu d'agrupar tota la lògica específica de cada vista. Tenen l'objectiu de gestionar cada una de les diferents pantalles que mostra l'aplicació. La lògica comú és heretada del *viewBaseController* i cada classe implementarà noves funcionalitats determinades per les accions que es podran efectuar en cada pantalla. Els controladors de les vistes són els següents:

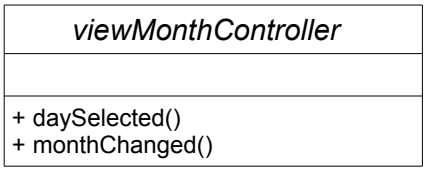
- viewListController: l'objectiu d'aquest controlador és gestionar la vista principal del mode llista. S'encarrega de demanar les cites que s'han de mostrar per pantalla al controlador *dataController*. El diagrama és el següent:



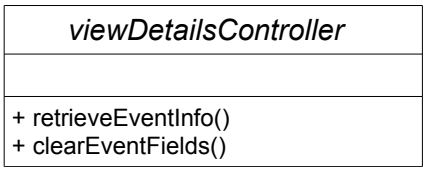
- viewDayController: aquest controlador té la finalitat de gestionar la vista principal del mode dia. S'encarrega de demanar les cites que s'han de visualitzar per pantalla al controlador *dataController*. El diagrama és el següent:



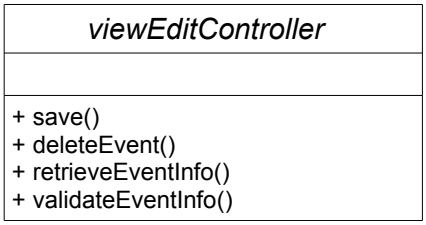
- viewMonthController: aquest controlador existeix per gestionar la vista principal del mode mes. S'encarrega de demanar les cites que s'han de visualitzar per pantalla al controlador *dataController*. Utilitza la classe *slenderCalendar* (controlador auxiliar) per mostrar un calendari per pantalla. El diagrama és el següent:



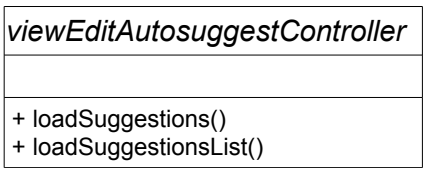
- viewDetailsController: es crea amb l'objectiu de gestionar la vista que mostra els detalls d'una cita. S'encarrega de demanar la informació de la cita que s'ha de visualitzar per pantalla al controlador *dataController*. El diagrama és el següent:



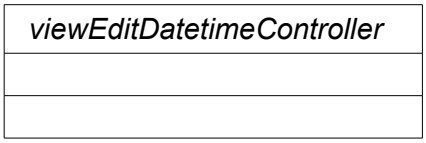
- viewEditController: aquest controlador es crea amb el propòsit gestionar la vista d'edició d'una cita. S'encarrega de demanar la informació de la cita que s'ha d'editar al controlador *dataController*. També s'encarrega de controlar quin element es vol editar, carregant una nova vista en cas necessari. També gestiona els esdeveniments de guardar o eliminar. El diagrama és el següent:



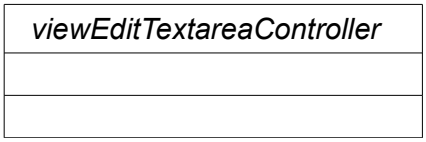
- viewEditAutosuggestController: es crea per gestionar la vista que mostra el widget que permet tota la lògica de l'auto suggeriment. Aquest controlador utilitza la classe slenderAutosuggest (un controlador auxiliar) per gestionar la part gràfica. Interacciona directament amb el servidor per demanar la informació que necessita. El diagrama és el següent:



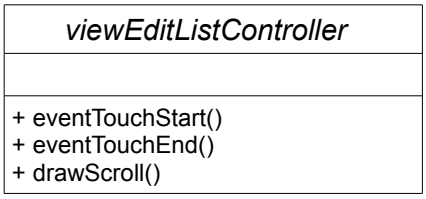
- viewEditDatetimeController: aquest controlador té la funció de gestionar la vista que mostra el widget que permet tota la lògica de la selecció de dates. El diagrama és el següent:



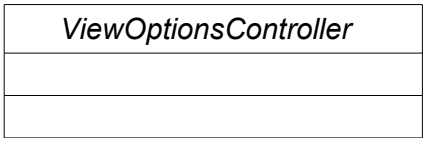
- viewEditTextareaController: aquest controlador té la funció de gestionar la vista que mostra l'eina per poder editar text. El diagrama és el següent:



- viewEditListController: aquest controlador s'encarrega de gestionar la vista que mostra una llista amb desplaçament vertical. El diagrama és el següent:



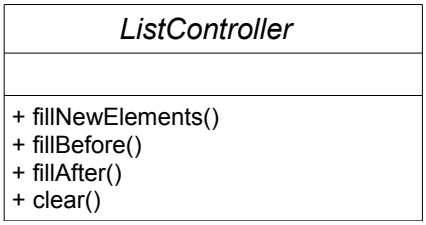
- ViewOptionsController: aquest controlador es crea amb la finalitat de gestionar la vista que mostra les opcions. El diagrama és el següent:



4.2.3.4 Controladors auxiliars

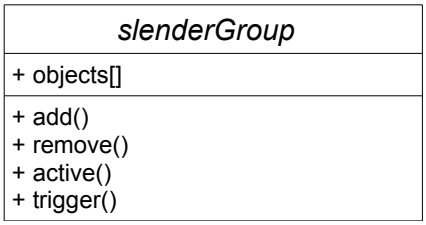
Els controladors auxiliars amb un propòsit específic són els següents:

- ListController: aquest controlador està creat per controlar la vista de les llistes. S'encarrega d'aportar els mètodes per treballar d'una manera còmoda amb les llistes, facilitant el poder afegir elements al principi, al final, etc. El diagrama és el següent:

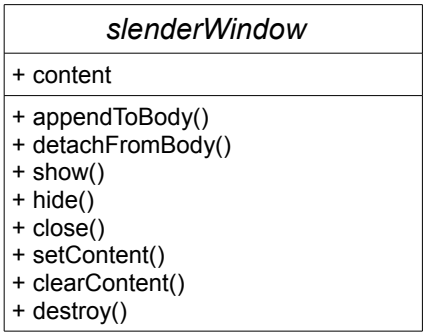


- slenderCalendar: aquest controlador es crea per gestionar el calendari que apareix en la vista de mes proporcionant tots els mètodes necessaris per poder interactuar amb el calendari. El diagrama és el següent:

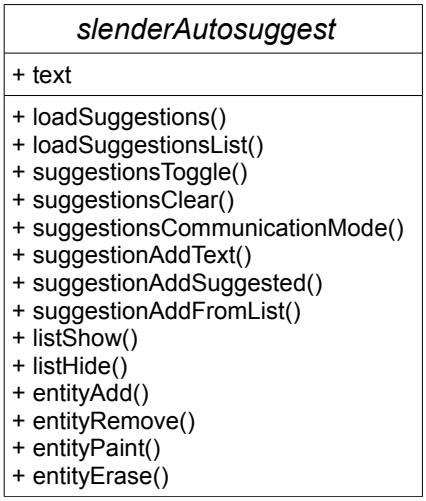
- slenderGroup: aquest controlador auxiliar apareix per permetre l'agrupació de botons. D'aquesta manera és possible controlar que només un botó estigui actiu al mateix temps dins del grup. El diagrama és el següent:



- slenderWindow: aquest controlador té la finalitat de permetre generar una finestra i carregar-hi qualsevol contingut. Permet diferents opcions de visualització. El diagrama és el següent:

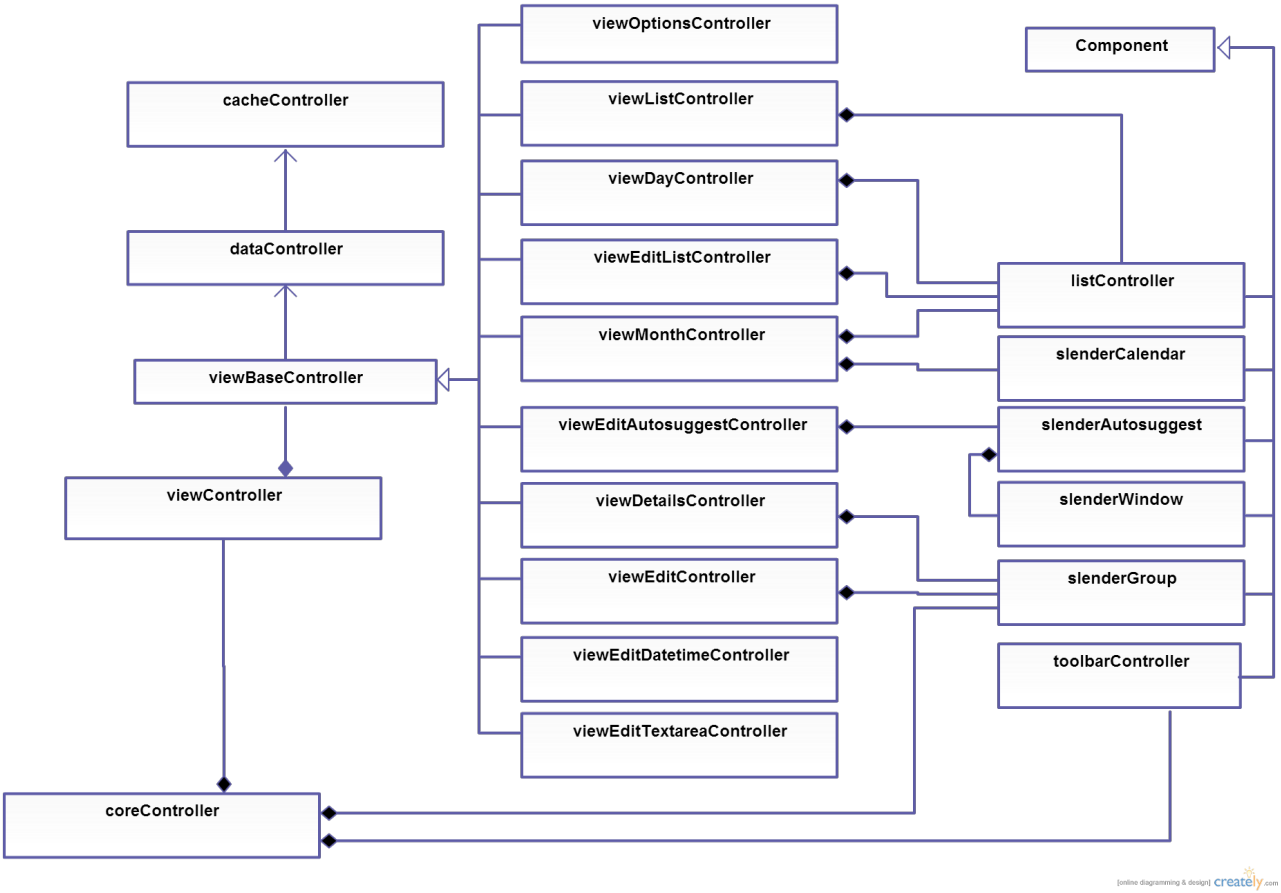


- slenderAutosuggest: aquest controlador té la funció d'implementar tota la lògica del widget d'auto suggeriment. Permet recuperar directament del servidor informació utilitzant els serveis habilitats per a tal finalitat. Amb aquesta informació recuperada en temps real es mostren suggeriments a mesura que l'usuari va introduint text. El diagrama és el següent:



4.2.4 Diagrama de classes

En el capítol anterior s'han introduït els controladors i les seves principals funcions. En aquest apartat es mostra la relació que existeix entre totes les classes que s'han definit amb el següent diagrama.



4.3 Nous serveis en el servidor

Un cop escollida la forma en com es transferirà la informació, ha fet falta ampliar alguns dels serveis del servidor per poder acceptar i transmetre informació en aquest nou format. Principalment s'ha dissenyat quatre serveis nous per atendre els següents tipus de peticions per poder recuperar cites:

- findByInterval: aquest servei espera una data d'inici i una data de fi, les quals indiquen l'interval sobre el que es desitja recuperar les cites.
- findById: aquest servei espera un identificador de cita per poder retornar la cita en qüestió.
- findSince: aquest servei espera una data d'inici i el número de dies amb cites que es vol recuperar.
- sync: aquest servei espera un conjunt de cites modificades per part del client i retorna el conjunt de cites que han sigut modificades des de la última sincronització. En aquestes cites retornades pel servidor s'inclouen les que s'han enviat en la petició de sincronització.

Juntament amb la informació particular de cada un dels serveis, totes les peticions porten adjunta informació sobre l'estat de la memòria cau del sistema. Aquesta informació permet al servidor saber quin és l'interval emmagatzemat a la memòria cau i en quina data ha sigut la última actualització de la memòria cau. Gràcies a aquesta informació, el servidor retorna en cada peticions les cites que han sigut actualitzades des de la última actualització.

Els tres primers serveix comparteixen característiques; es fa una petició temporal i es retorna la informació demanada més les últimes actualitzacions. En canvi, el servei *sync* permet enviar al servidor un conjunt de cites perquè s'actualitzin, en resposta es retorna el mateix que en els altres serveis: totes les cites modificades des de la última actualització.

Complementàriament s'ha dissenyat també un servei per atendre les peticions del camp d'auto suggeriment. Aquest servei ha de permetre escoltar directament cada un dels diferents camps i respondre amb una llista dels resultats que compleixen els paràmetres que han sigut enviats. El servei hauria d'atendre peticions per als següents camps: assignats, assistents, clients, contactes, anunciants, objectius i mitjans. En aquestes peticions s'hi accepten els següents paràmetres:

- suggest: és el text que introdueix l'usuari.
- exclude: un conjunt d'identificadors que representen els elements que ja formen part de la selecció. Això permet al servidor no enviar elements repetits.
- amount: aquest valor ens permet indicar quants elements s'han de retornar que coincideixin amb el text enviat. Aquest paràmetre està pensat per poder recuperar un únic suggeriment quan s'introdueix el text i després poder recuperar una llista sencera quan s'entra en el mode llista de suggeriment.
- filters: és paràmetre extra que permet passar informació addicional per si s'ha de realitzar algun filtre extra en funció d'alguna altra informació. En concret s'ha fet servir per filtrar els contactes en funció del client seleccionat.

A totes aquestes peticions el servidor sempre respon amb una llista de tants elements com s'ha especificat en el paràmetre *amount* i amb un altre paràmetre que indica el número de coincidències totals.

4.4 Disseny de la interfície d'usuari

La interfície de l'usuari es pot considerar un dels punts més importants de l'aplicació. El disseny ha de presentar una solució amigable amb l'usuari, fàcil d'utilitzar i que s'ajusti a totes les necessitats sense mostrar més opcions que les indispensables per al correcte desenvolupament de totes les funcions.

En el moment de dissenyar la interfície s'ha tingut present en tot moment que seria utilitzada des de dispositius mòbils amb una pantalla de reduïdes proporcions. Per aquest motiu es va optar per imitar les interfícies més comuns tant en IOS com en Android, per assegurar-nos un disseny elegant, robust i sobretot funcional.

L'apartat gràfic s'ha distribuït en blocs que s'anomenaran vistes. Així, per a cadascuna de les possibles pantalles que es mostren en l'aplicació s'ha creat un ***controlador vista*** que s'encarrega de gestionar tota la logística d'aquell apartat.

Existeixen tres vistes que són les principals de l'aplicació. Aquestes vistes permeten veure les nostres cites, però en 3 formats diferents. Aquestes vistes reben el nom de ***Llista***, ***Dia***, ***Mes***. Existeixen dues vistes més auxiliars que s'han creat per a temes logístics fora del principal funcionament de l'aplicació. Aquestes vistes són les ***Opcions***, i la vista d'***Autentificació***.

Un cop s'accedeix a una cita, apareix un altra conjunt de vistes encarregades de gestionar totes les operacions que es poden fer sobre aquesta. D'entrada, quan s'edita una cita, s'accedeix a la vista ***Detall***. Des de detall es pot accedir a la vista ***Editar***, la qual, si decidim crear una cita nova, és la primera que apareix.

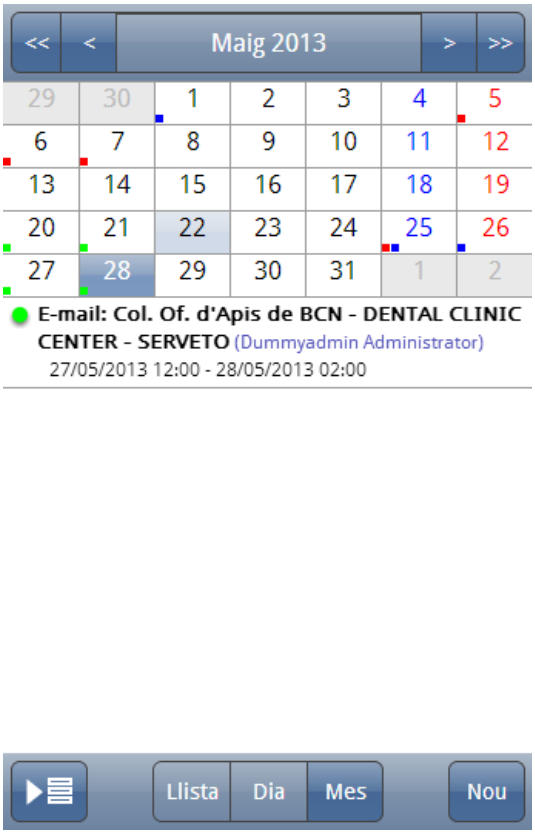
Un cop tenim oberta la vista Editar, aquesta ens permet accedir a tots els camps que siguin modificables sobre aquella cita. En funció del tipus de camp que vulguem editar, la vista ens donarà pas a cadascuna de les vistes dissenyades en funció del que calgui editar. Aquestes vistes d'edició són: vista ***Edició de temps***, vista ***Edició de text***, i vista ***Edició amb suggeriment***. Per editar un auto desplegable s'utilitza la funció nativa del navegador. Internament existeix una vista auxiliar que és la ***Selecció de Llista***. Aquesta apareix en el moment de mostrar una llista d'on es pot seleccionar algun element.

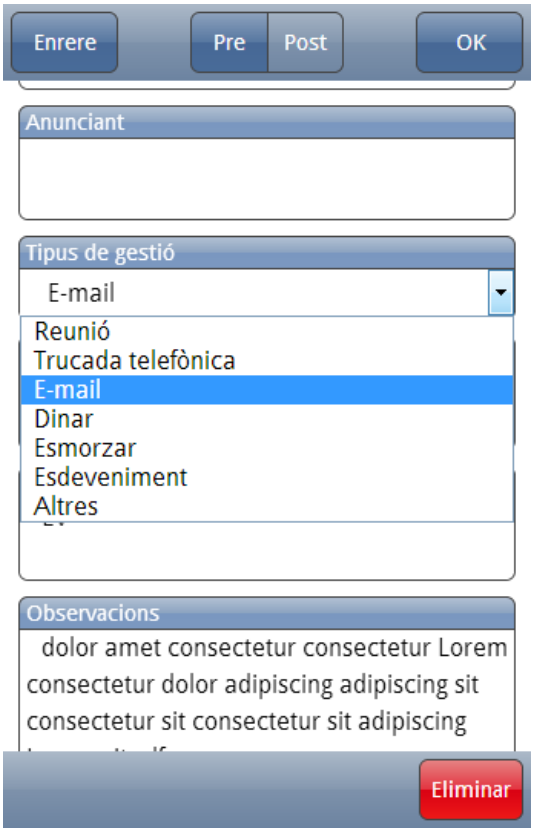
4.4.1 Disseny d'usabilitat

Per al disseny de la interfície es fa agafar com a referència el disseny presentat pel calendari de iOS. Apple ja va fer en el seu moment un estudi d'usabilitat i les conclusions que van treure estan plasmades en cadascuna de les aplicacions que publiquen. Totes elles mostren un disseny molt similar en quant a distribució dels elements interactius. Així doncs, si es navega una mica per les opcions de configuració del sistema iOS o les aplicacions natives que venen amb el

sistema, de seguida es pot veure que l'opció més bona és situar els botons en la part inferior o superior de la pantalla agrupats en unes barres que marquen els límits entre l'acció principal i les accions secundaries.

En aquestes dues imatges es pot veure la distribució dels elements dins la interfície. En aquests casos concrets es pot veure que existeixen barres tant a la part superior com a la part inferior.





Estudiant la interfície del sistema iOS no es va observar cap distribució concreta dels elements dins les barres superiors o inferiors. Per tant la forma en com els botons es van distribuir en les barres es va fer amb els nostres criteris propis d'usabilitat; Basant-nos en que la majoria d'usuaris utilitzen el polze de la mà dreta per realitzar la majoria d'operacions es van situar els botons amb accions més crítiques en zones de més difícil accés per al dit. Les accions més comuns es van situar en les parts de més fàcil accés. Així doncs tenim que l'acció d'eliminar una cita ocupa la zona on és més complicat arribar amb el dit. En canvi, les accions d'acceptar o fer enrere estan en la zona més accessible de la pantalla ja que és el més probable que vulgui fer l'usuari.

Capítol 5

Implementació de l'aplicació

En aquest apartat s'expliquen les parts més interessants en quant a la implementació de les tècniques utilitzades en l'aplicació.

5.1 Indexat de les cites

Aquest va ser un dels primers objectius en quan a la lògica de l'aplicació. Era necessari mantenir les cites de forma ordenada per poder-les recuperar àgilment sense necessitar examinar-les totes cada vegada que es demanava un subgrup.

Les situacions en que era necessari recuperar conjunts de cites era en les tres vistes principals. La vista de llista mostra les cites en una llista en grups de dies, és a dir, quan es mostra un dia, es mostren totes les cites d'aquell dia. En la vista de dia es mostren totes les cites d'un dia. Finalment, en la vista de mes, es mostren les cites d'un dia, però internament es recuperen les cites de tot el mes. Analitzant aquesta situació es va detectar que la unitat mínima que es recuperava sempre era el dia. Si es volia recuperar menys que un dia havia de ser una única cita especificant el seu identificador.

Atenent les necessitats del punt anterior es va decidir implementar dos índexs per poder mantenir una doble ordenació de les cites. A l'hora d'implementar aquests dos índex dins la classe *cacheController* es va optar per fer servir la classe *Objecte* de *JavaScript*. *JavaScript* implementa l'Objecte com si d'un mapa es tractés. Permet associar una cadena de text a qualsevol dels tipus bàsics de *JavaScript* (cadena, numèric, vector, objecte, etc.).

```
eventsById_ = {};
eventsByDay_ = {};
```

En el cas de l'indexat per identificador no era aconsellable utilitzar un vector ja que emmagatzemar una cita amb l'identificador 8000, per exemple, implicava la declaració d'un vector de 8001 posicions, on molt probablement la gran majoria de posicions intermèdies no tindrien cap dada emmagatzemada. En benefici de la memòria es va sacrificar una mica de velocitat en l'accés a la dada i es va optar per utilitzar el mapa.

En el cas de l'indexat per dies es va decidir crear un vector per cada dia que tingués una cita. Aquests vectors queden guardats dintre del mapa *eventsByDay* utilitzant com a clau la cadena que representa el segon zero del dia representat en Unix Timestamp[4]. Per exemple 1370815200000 representa les 0 hores, 0 minuts, 0 segons i 0 mil·lisegons del 10 de juny de 2013. Així doncs, per indexar les cites es mira la data d'inici i la data de fi d'una cita. Si la cita transcorre total o parcialment en un dia, es busca en el mapa si el dia existeix (utilitzant la forma explicada per saber la cadena representant d'un dia). En cas negatiu es crea un vector i s'afegeix a l'índex amb la clau corresponent. Un cop tenim el vector el recorrem buscant si hi ha alguna cita que comenci més tard que la que tenim. Si la trobem inserim en el vector davant d'aquesta cita que estem indexant. En cas contrari l'afegim al final del vector.

Després d'efectuar diverses vegades aquesta operació ens queda que l'objecte EventsByDay té la següent estructura:

```
eventsByDay = {
        1370815200000: Array[1],
        1370901600000: Array[1],
        1371074400000: Array[2],
        1371160800000: Array[3],
        1373320800000: Array[1],
        1373407200000: Array[1]
 }
```

Dins de cada vector podem trobar les cites del dia en qüestió ordenades de forma que la que comença abans en el temps es situa en la primera posició i la més nova en la última.

Aquestes accions són dutes a terme pel mètode *addEvent* de la classe *cacheController*. Primer afegeix les cites en el mapa *eventsById* i després realitza totes les comparacions per indexar de forma correcta la cita en el mapa *eventsByDay*.

5.2 Memòria cau del navegador

El fet de treballar amb la memòria cau que proporciona el navegador a partir de la versió 5 de l'*HTML* i veure les limitacions que tenia va causar que s'haguessin de generar uns mètodes encarregats de convertir la informació d'una cita en una cadena de text i que després fossin capaços de recuperar una cadena de text i restaurar-la a una cita.

El problema radica en que la memòria cau del navegador només és capaç de guardar parelles de cadenes de text. Per facilitar aquesta tasca *JavaScript* disposa d'una funció que es crida de la següent manera:

4 Temps Unix es defineix com la quantitat de segons transcorreguts des la mitjanit de l'1 de gener de 1970.

```
JSON.stringify( objecte a convertir en text );
```

Aquest mètode retorna l'objecte convertit en una seqüència JSON. D'aquesta manera és possible emmagatzemar-lo dins la memòria local. El codi que realitza la següent acció és:

```
dump : function() {
         if ( localStorage ) {
                localStorage.setItem( "events", JSON.stringify( this.eventsById_ ) );
                localStorage.setItem( "cacheStart", this.cacheStart.getTime() );
                localStorage.setItem( "cacheEnd", this.cacheEnd.getTime() );
                localStorage.setItem( "cacheLastUpdate", this.cacheLastUpdate.getTime() );
         }
 }
```

En aquest tros de codi es pot veure com s'emmagatzemen quatre variables a la memòria. És necessari notar que l'índex que s'agafa per convertir a text és el que guarda les cites per l'identificador i no pels dies. Això és perquè en aquest índex cada cita es guarda una única vegada i totes en l'objecte principal. No seria correcte realitzar aquesta acció sobre l'índex que guarda les cites agrupades en dies, perquè tindríem una conversió a text amb molta informació duplicada.

Existeixen dos inconvenients al realitzar el procés de conversió a text; tots els objectes són passats a text. Això vol dir que on abans es tenia un objecte amb atributs interns i certa complexitat, si no s'ha implementat bé la funció *toString()* de l'objecte, ens podem trobar amb una cadena buida.

L'altra inconvenient apareix quan es vol realitzar el procés invers. No resulta possible restaurar objectes que no són dels tipus bàsics. Això exclou inclús els objectes de tipus Date.

Per causa d'aquests problemes no és possible restaurar les cites tal com van ser salvades. Així doncs s'ha implementat un algoritme que restaura les cites en un vector i a continuació recorre el vector afegint totes les cites a la memòria cau de l'aplicació utilitzant el mètode creat per realitzar aquesta acció.

El tros de codi que restaura les cites en un vector és el següent:

```
eventsStack = JSON.parse( localStorage.getItem("events") );
```

EventsStack és un vector que guarda totes les cites com ho fa la variable eventsById. La diferència és que aquestes cites són incorrectes per culpa dels camps temporals que de moment són cadenes de text. Quan es crida la funció *addEvent* amb la cita com a paràmetre, el primer que realitza aquest mètode és cridar un altra mètode que es diu *correctEvent*. Aquest últim s'encarrega de corregir les cites i deixar-les en un estar correcte.

El codi de la funció que corregeix les cites és el següent:

```
correctEvent : function( _event ) {
        if ( typeof( _event.id ) != 'Number' ) {
                _event.id = Number( _event.id );
        }
        if ( typeof( _event.startDate ) != 'object' ) {
                _event.startDate = new Date( Number( _event.startDate ) );
        }
        if ( typeof( _event.endDate ) != 'object' ) {
                _event.endDate = new Date( Number( _event.endDate ) );
        }
        if ( typeof( _event.lastUpdate ) != 'object' ) {
                _event.lastUpdate = new Date( Number( _event.lastUpdate ) );
        }
}
```

Com es pot veure en el codi, la funció comprova els quatre camps crítics després de la conversió de text a objecte. Si els camps no són del tipus que els pertoca es realitza la conversió necessària per deixar-los en l'estat correcte.

5.3 Recuperació de cites per interval

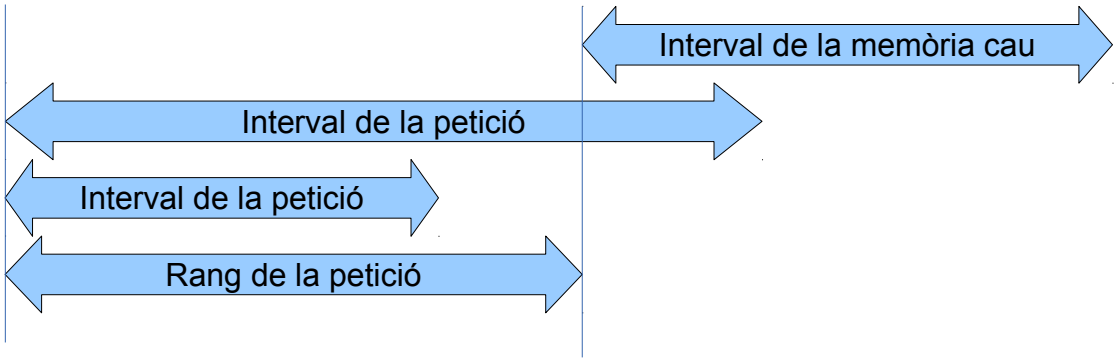
El fet de recuperar cites implica utilitzar la classe *dataController*. Aquesta classe incorpora un conjunt de mètodes de per agrupar en simples crides tot el procés que comporta la recuperació de cites.

Primer cal explicar el mètode que es segueix en totes les peticions de recuperació de cites. La classe *cacheController* té dues propietats que indiquen la data més baixa i la data més alta que es guarden en la memòria cau de l'aplicació. Aquestes dues dates són importants per determinar l'interval temporal emmagatzemat en la memòria local i poder detectar quan una petició demana cites que estan fora de la memòria cau.

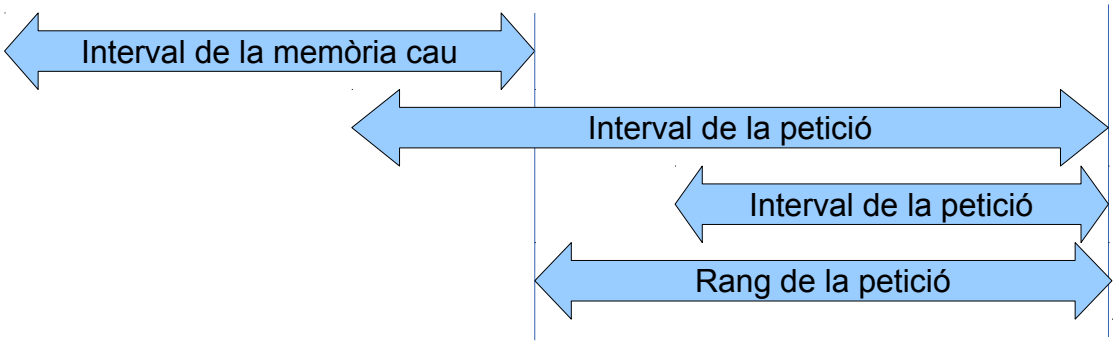
Feta la breu explicació del concepte que es segueix per determinar d'on surten les cites es defineix l'algorisme del mètode *findByInterval* de la següent manera:

- Es rep una petició on es demanen totes les cites que pertanyen a un interval.
- Es comprova si l'inici de l'interval està comprès entre l'inici i el final de l'interval emmagatzemat en la memòria cau. En cas afirmatiu es recuperen les cites que estan dins l'interval i es retornen.
- En cas que part de l'interval que es demana o la totalitat d'aquest estigui fora del rang contingut em la memòria local, es faran les peticions necessàries al servidor per recuperar els intervals necessaris. Hi ha tres possibles situacions:

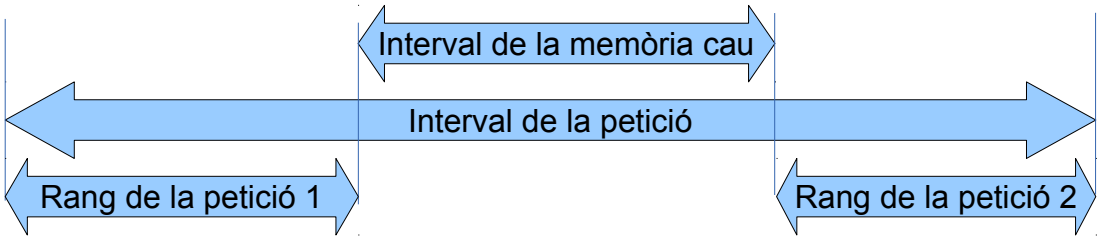
- Ens demanen un interval que està totalment o parcialment fora de l'inici de la memòria cau. En aquest cas es demanarà al servidor des de la data d'inici del rang que es demana fins la data d'inici de la memòria cau.



- Ens demanen un interval que està totalment o parcialment fora del final de la memòria cau. En aquest cas es demanarà al servidor des de la data de fi de la memòria cau fins la data de fi del rang que es demana.



- Ens demanen un interval que sobrepassa la data d'inici i la data de fi de l'interval contingut en la memòria cau. En aquest es tractarà la crida com si fos el primer cas realitzant la petició corresponent. Un cop finalitzada la recepció de les cites el sistema torna a efectuar aquesta crida amb els mateixos paràmetres. La segona vegada que s'entra a la crida es desemboca en el segon cas. D'aquesta manera queden coberts tots els casos.



- Un cop completada la petició al servidor es torna a cridar la funció amb els paràmetres originals per realitzar una segona petició en el tercer dels casos. Per controlar el nombre de vegades que es demanen intervals al servidor s'inclou una marca en els paràmetres

originals. Aquesta marca permet que si ja s'han fet dues peticions com a màxim i encara, pel motiu que sigui, no es té l'interval complet, no es segueixin fent peticions de forma indefinida.

Aquest mètode permet un paràmetre que especifica si les cites que s'han de retornar es volen agrupades per dies o sense cap agrupació. En el segon cas es retorna un vector no ordenat. En el primer cas es retorna un mapa de vectors que segueix la mateixa estructura i característiques que la propietat *eventsByDay* de la classe *cacheController*.

5.4 Recuperació de cites per quantitat de dies

El mètode *findBySince* permet recuperar un número concret de dies no consecutius amb cites. És a dir, només té en compte els dies amb cites per calcular els dies que ha de retornar. Aquest mètode es va implementar per poder simular un comportament similar al del calendari natiu d'Android.

Aquest algoritme ha sigut fàcil d'implementar gràcies a que existeix un índex que agrupa les cites per dies. D'aquesta forma la única funció que ha de realitzar l'algorisme és consultar a partir de la data proporcionada si existeixen prou dies amb cites.

Existeix una particularitat en el funcionament que fa que recuperar les cites no sigui tan ràpid com s'esperava. Els dies amb cites estan indexats utilitzant un mapa. Malgrat que les proves realitzades apunten a que l'objecte natiu JavaScript, que es comporta com un mapa en quan a la creació d'atributs es refereix, guarda els atributs amb un cert ordre. Com que aquesta particularitat no forma part de l'especificació es va decidir ignorar aquesta característica i implementar un sistema per processar el mapa que guarda els dies de les cites (*cacheController.eventsByDay*).

La forma en com treballa l'algorisme és senzilla; es crea un índex que compleix amb les característiques descrites per als índex dels dies. Aquest índex correspon al dia a partir del qual s'ha de cercar. Mentre l'índex estigui dins l'interval de la memòria cau es va comprovant si existeix aquest índex en el mapa *cacheController.eventsByDay.* Si existeix es guarden les cites d'aquest dia en un mapa indexat per l'identificador de les cites, es compte que ja s'ha trobat un dia, s'augmenta l'índex fins apuntar al dia següent i es repeteix el procés.

Hi haurà moltes cites que potser ocupin més d'un dia i per tant estiguin repetides a molts dies. Recuperant les cites i agrupant-les en un mapa indexat per l'identificador de cada cita s'evita tenir cites duplicades.

En cas que l'algorisme arribi a la fi de la memòria cau i encara no tingui el número de dies demanat, es farà una petició al servidor per recuperar més cites. Aquesta petició cridarà a un mètode idèntic implementat en la part del servidor que intentarà retornar una quantitat de cites a partir de la data demanada suficients per omplir N dies diferents.

La lògica de la comunicació farà que si la data d'inici està fora de l'interval es retornaran també totes les cites compreses entre aquesta data i la fi o inici de l'interval de la memòria local (depenent de si la data d'inici està abans o després de l'interval). Amb aquest sistema s'assegura una continuïtat entre l'inici i la fi de la memòria cau i de la primera fins la última cita.

Un cop rebuda la resposta del servidor es torna a cridar a la funció amb els paràmetres originals per intentar una segona vegada completar el volum demanat de dies. Aquesta segona vegada, en els paràmetres originals també hi ve una marca que indica que ja s'ha realitzat una petició al servidor. En aquest mètode com a màxim es permet una crida. Si després de la crida encara no és possible retornar cites com per omplir la quantitat de dies demanats, llavors es retornarà la quantitat de cites de les que es disposi i es donarà per finalitzada la petició.

5.5 Comunicació amb el servidor

Per poder realitzar la comunicació ha fet falta implementar els mètodes descrits en l'apartat de "Nous serveis en el servidor" (Capítol 4, apartat 5). La forma en com es criden aquests mètodes és realitzant peticions *HTTP* utilitzant les habilitats asíncrones del navegador web. Per poder realitzar aquestes crides el navegador disposa d'un objecte anomenat XHR[5] que té uns mètodes que permeten invocar crides *HTTP* sobre una URI[6] específica. Per temes de seguretat aquesta URI ha de pertànyer al mateix domini sota el que funciona l'aplicació.

Quan es realitza una petició HTTP sobre una URI específica el servidor captura la petició i la redirigeix al controlador definit per atendre específicament aquella crida. Per determinar si una petició encaixa en algun dels controladors definits per atendre les crides es comproven tots els paràmetres d'entrada juntament amb la URI.

En el nostre cas els serveis estaven configurats per atendre una URI específica i un conjunt de dades formatat utilitzant JSON. A demés, aquestes dades enviades són interpretades i s'intenten convertir en una classe vàlida de Java si el mètode així ho requereix. Això pot causar que si algun dels paràmetres enviats al servidor no és correcte o falta la petició sigui rebutjada.

Tots els nous serveis implementats en el servidor es poden agrupar en els que serveixen cites i els que serveixen informació als camps d'una cita. Per al primer tipus es va definir un format base i uns petits camps que anaven variant en funció del servei cridat. Per al segon servei es va definir un únic tipus de dades diferenciant exclusivament el servei per la URI que es feia servir en la crida.

5 XMLHttpRequest
6 Universal Resource Identifier. És més complet que el terme URL perquè incorpora aquest juntament amb altra informació com els paràmetres de consulta (?) o fragment (#)

5.5.1 Petició de cites

Per poder realitzar les peticions de cites s'han implementat en la part del servidor els següents serveis en les següents URIs.

- FindById --> mobile/findById
- findByInterval --> mobile/findByInterval
- findSince --> mobile/findSince
- Sync --> mobile/sync

Per totes aquestes URIs s'ha de configurar la petició com amb el tipus POST i el tipus de dades que s'enviaran com JSON. D'aquesta manera quan es fa la petició s'envien les capçaleres correctes i la informació enviada com a POST és interpretada correctament com una estructura JSON.

En el moment d'implementar els serveis es va pactar amb el seu desenvolupador un format per poder transmetre tota la informació necessària en cada crida. Aquest format surt de convertir a JSON un objecte JavaScript; aquest objecte rep el nom de DTO[7]. Es va acordar que en cada crida sempre s'enviarien dos tipus de camps; per una banda s'enviarien un conjunt de camps fixats que farien referència a l'estat de la memòria cau de l'aplicació i que contindrien totes les cites modificades per poder informar al servidor dels canvis realitzats. Per l'altra banda s'enviarien els camps específics de cada tipus de petició.

El DTO base amb els camps que sempre s'envien és el següent:

```
DTO {
        cacheEnd: 1371679199999,
        cacheLastUpdate: 1371004967180,
        cacheStart: 1370037600000,
        meetings: null
}
```

Els camps que s'envien en totes les peticions de recuperació de cites són els següents:

- cacheStart: indica l'inici de l'interval de la memòria cau local.
- cacheEnd: indica la fi de l'interval de la memòria cau local.
- cacheLastUpdate: indica en quin moment es va fer la última sincronització de la memòria local amb el servidor.
- meetings: és un vector on s'envien totes les cites que han sofert algun canvi i que aquest s'ha de comunicar al servidor.

A continuació es mostren els DTO dissenyats per transmetre la informació de cada una de les peticions.

7 Data Transfer Object

- FindByInterval: en aquesta crida s'estenen els camps fixos amb els camps startDate i endDate. Aquests dos camps serveixen per indicar quin interval de cites es vol recuperar.

```
DTO {
        cacheEnd: 1371679199999,
        cacheLastUpdate: 1371004967180,
        cacheStart: 1370037600000,
        meetings: null,
        startDate: 1371592800000,
        endDate: 1372629599999
}
```

- FindSince: en aquesta crida s'estenen els camps fixos amb els camps startDate i amount. Aquests dos camps serveixen per indicar a partir de quina data s'ha de buscar la quantitat de dies desitjada.

```
DTO {
        cacheEnd: 1372629599999,
        cacheLastUpdate: 1371006089178,
        cacheStart: 1370037600000,
        meetings: null,
        startDate: 1370988000000,
        amount: 8
}
```

- findById: en aquesta crida s'estenen els camps fixos amb el camp id. Aquest camp especifica l'identificador de la cita que es vol recuperar.

```
DTO {
        cacheEnd: 1372629599999,
        cacheLastUpdate: 1371006089178,
        cacheStart: 1370037600000,
        meetings: null,
        id: "12113"
}
```

- Sync: en aquesta crida s'envien els camps fixos sense afegir-ne cap altra. Serveix per sincronitzar l'interval local amb les actualitzacions que pugui tenir el servidor.

En qualsevol de les quatre crides definides, la resposta del servidor sempre és la mateixa; un vector que conté totes les cites que s'han de transmetre cap al client. Aquestes cites corresponen a les cites resultants de la petició i a totes les cites que han sigut modificades des de la última petició al servidor. D'aquesta manera s'aprofiten totes les crides als serveis per mantenir sincronitzada l'aplicació amb el servidor.

5.5.2 Petició de suggeriments

Per poder realitzar les peticions dels suggeriments s'han implementat en la part del servidor els següents serveis en les següents URIs.

- Assignats --> mobile/suggest/assignees
- Assistents --> mobile/suggest/attendees
- Clients --> mobile/suggest/clients
- Contactes --> mobile/suggest/contacts
- Anunciants --> mobile/suggest/advertisers
- Objectius --> mobile/suggest/milestones
- Mitjans --> mobile/suggest/mediums

Per realitzar les peticions s'utilitza el següent DTO acordat amb el desenvolupador dels serveis. En l'apartat de "Nous serveis en el servidor" (Capítol 4, apartat 5) es descriu quina funció te cada camp:

```
DTO {
        amount: 1,
        exclude: [2],
        suggest: "a",
        filters: []
}
```

El servidor sempre respondrà amb la mateixa estructura de dades:

```
{
        entities: [
                { name:Ander Anitua Arteche, id:8567, extraInfo:null }
        ]
        matches: 16
}
```

Els camps de la resposta són els següents:

- Entities: és un vector que conté tants suggeriments com s'hagin especificat en el camp amount de la crida.
- Matches: és la quantitat de resultats que s'han trobat que encaixen amb la cadena enviada.

El motiu pel qual es retorna la quantitat total d'elements que s'han trobat és perquè el camp d'auto suggeriment té dues etapes. En la primera només es mostra un suggeriment, i si el total de suggeriments és més gran que zero i més petit o igual a cinquanta, existeix la possibilitat d'accedir a una vista on es mostra una llista amb tots els suggeriments. Per realitzar aquesta acció es torna a fer la mateixa petició al servidor però canviant el valor que indica la quantitat d'elements que volem des de 1 al valor retornat en el camp *matches*.

Capítol 6

Implantació

En aquest capítol es vol presentar tot el procés que s'ha dut a terme durant el procés de publicació de l'aplicació. Això inclou les proves amb els usuaris, el desplegament, les formacions als usuaris, les diferents configuracions realitzades als terminals i el seguiment posterior que es va dur de l'ús de l'aplicació.

6.1 Proves UAT

Les proves que es van realitzar amb usuaris van començar després de finalitzar l'etapa d'implementació. Durant tres setmanes es va agafar un petit grup de cinc comercials de la central de Publipress Media a Barcelona (per temes de proximitat en cas d'haver de fer alguna reunió). Se'ls va fer una petita formació de quatre hores i se'ls va donar el dispositiu Samsung Galaxy II que finalment es repartiria als demés comercials.

Se'ls va donar una petita documentació on s'explicaven totes les característiques que permetia fer l'aplicació i on es descrivien un conjunt d'accions i els seus resultats. Es va demanar que durant tres setmanes anessin provant totes les operacions descrites i comprovant que en totes les situacions els resultats eren els esperats. Aquesta petita documentació representava el conjunt de proves UAT[8]. El conjunt de proves va ser extret dels requeriments i els casos d'us especificats per a l'aplicació.

Durant aquest període de tres setmanes només hi va haver una petició referent a les facilitats de l'editor de comentaris d'una cita; una comercial ens va comentar que ella escrivia les notes de les reunions directament sobre l'aplicació mòbil. El problema sorgia que si per algun motiu es tancava el navegador o es penjava el mòbil tot el que havia escrit es perdia. En va demanar doncs si es podia d'alguna manera posar algun botó per salvar sense haver de sortir del mode edició. Per aquest motiu es va dissenyar la vista d'edició de text amb un botó que permet guardar sense abandonar la vista.

8 User Acceptance Test

6.2 Desplegament

Un cop superades les tres setmanes de proves es va realitzar el desplegament final de l'aplicació. En aquesta etapa només es va realitzar una acció: distribuir els dispositius mòbils a tots els comercials de Publipress Media amb les configuracions i accessos directes adients per facilitar l'accés a l'aplicació. No es va realitzar cap altra acció perquè l'aplicació ja s'hi podia accedir de forma pública i només quedava donar llum verda per a que els comercials comencessin a treballar amb ella.

6.2.1 Formació

Relacionat amb l'etapa de desplegament està el tema de la formació. Durant tota una setmana a Barcelona es van anar formant els diferents equips comercials i entregant-los la documentació que prèviament s'havia donat al grup pilot. La formació a Madrid es va dur a terme en un únic dia perquè el grup de comercials allà present és més reduït.

6.2.2 Configuració

Tots els dispositius Samsung Galaxy II van passar per les mans del departament de *Helpdesk*, els quals van instal·lar la versió mòbil del navegador Chrome i van crear els corresponents accessos directes i entrades en els marcadors per facilitar l'accés dels comercials a l'aplicació.

6.3 Seguiment

Un cop fet el desplegament i durant els següent mes s'ha anat fent un seguiment de l'ús i possibles errors de l'aplicació. Es controlava la quantitat de vegades que els comercials realitzaven diferents tasques des de l'aplicació mòbil. Tot aquest seguiment es podia fer gràcies a que la part del servidor guardava l'accés que es feia a totes les adreces web (tant de l'aplicació d'escriptori com l'aplicació mòbil).

Es va poder observar com inicialment l'ús era escàs i com poc a poc l'ús va anar creixent de forma constant. Un cop superat el mes de supervisió s'ha comprovat que el 85% del comercials fan ús de l'aplicació de forma continuada o esporàdica.

Durant aquest període de seguiment també s'esperava que sorgissin errors de funcionament o altres incidències, però no s'ha reportat cap mal funcionament per part dels comercials i tot sembla que funciona de forma correcte.

Capítol 7

Estudi econòmic i planificació

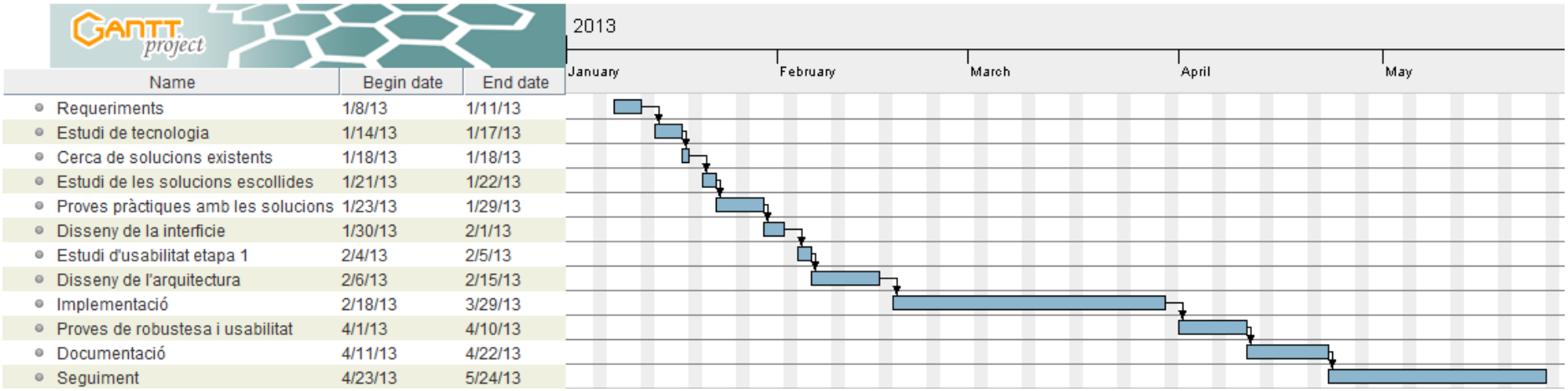
Inicialment el projecte va ser plantejat per a un total de 400 hores i el pressupost era de 8000€. La planificació es va dividir en les següents etapes:

- Requeriments
- Estudi de tecnologia
- Cerca de solucions existents
- Estudi de les solucions escollides
- Proves pràctiques amb les solucions
- Disseny de la interfície
- Estudi d'usabilitat
- Disseny de l'arquitectura
- Implementació
- Proves de robustesa i usabilitat
- Documentació
- Seguiment

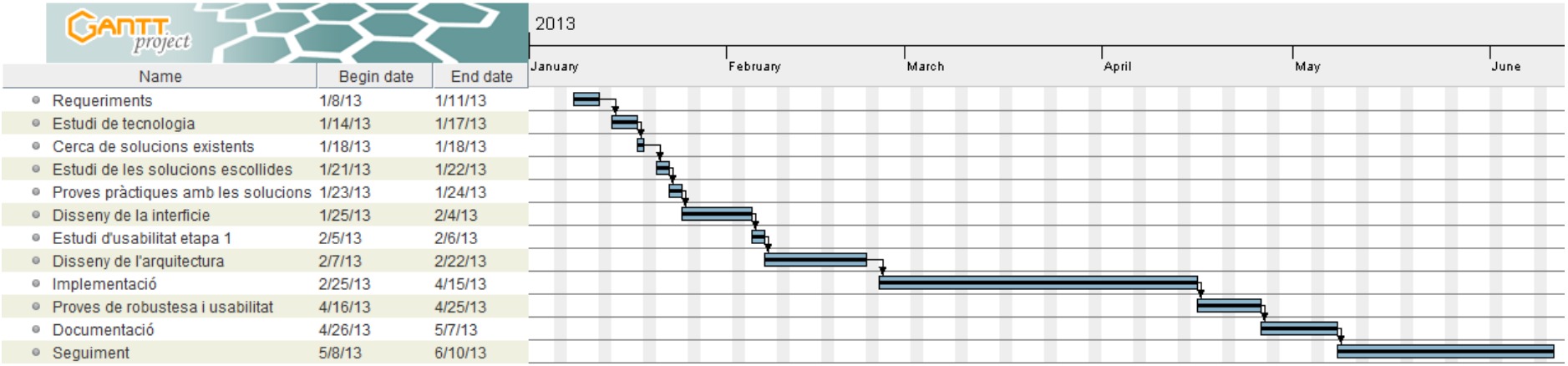
Les etapes planificades no van canviar durant el desenvolupament de l'aplicació malgrat que si es van entre mesclar. Un cop finalitzada l'aplicació es va fer una recapitulació del temps invertit en cada etapa i es va refer el diagrama de Gantt per poder comparar en quins punts s'havia efectuat la desviació inicial.

7.1 Previsió i desviació

El projecte estava previst que tindria una durada de 67 dies en jornades de 6 hores (402 hores). Finalment el projecte es va allargar fins als 79 dies amb un global de 474 hores. En els dos diagrames de Gantt següents es poden veure les petites diferències.



Name	Begin date	End date
Requeriments	1/8/13	1/11/13
Estudi de tecnologia	1/14/13	1/17/13
Cerca de solucions existents	1/18/13	1/18/13
Estudi de les solucions escollides	1/21/13	1/22/13
Proves pràctiques amb les solucions	1/23/13	1/29/13
Disseny de la interficie	1/30/13	2/1/13
Estudi d'usabilitat etapa 1	2/4/13	2/5/13
Disseny de l'arquitectura	2/6/13	2/15/13
Implementació	2/18/13	3/29/13
Proves de robustesa i usabilitat	4/1/13	4/10/13
Documentació	4/11/13	4/22/13
Seguiment	4/23/13	5/24/13



Name	Begin date	End date
Requeriments	1/8/13	1/11/13
Estudi de tecnologia	1/14/13	1/17/13
Cerca de solucions existents	1/18/13	1/18/13
Estudi de les solucions escollides	1/21/13	1/22/13
Proves pràctiques amb les solucions	1/23/13	1/24/13
Disseny de la interficie	1/25/13	2/4/13
Estudi d'usabilitat etapa 1	2/5/13	2/6/13
Disseny de l'arquitectura	2/7/13	2/22/13
Implementació	2/25/13	4/15/13
Proves de robustesa i usabilitat	4/16/13	4/25/13
Documentació	4/26/13	5/7/13
Seguiment	5/8/13	6/10/13

Si fem un petit estudi sobre la distribució de les fases podem observar que la major desviació es va produir en la fase d'implementació, allargant quasi vuit dies el temps de desenvolupament respecte els 40 dies previstos. Això suposa una desviació del 20% sobre el previst inicialment. El motiu d'aquesta desviació ha sigut l'increment de complexitat a l'hora d'implementar tots els controladors. Han aparegut noves necessitats que no s'havien tingut en compte i han fet que el desenvolupament s'allargués. També van sorgir alguns imprevistos a l'hora d'implementar la comunicació entre el client i el servidor, cosa que va incrementar algunes hores més el procés d'implementació.

Després també es pot detectar una desviació en el disseny de la interfície. S'ha passat de 18 hores a les 42 hores i això ha suposat un desviament d'un 233% sobre el càlcul inicial. Un cop més, el disseny de la interfície no ha sigut trivial i s'ha posat un esforç important en crear un producte que fos intuïtiu i còmoda d'usar.

La tercera desviació s'ha produït en la fase de disseny de l'arquitectura. Inicialment s'havia calculat una durada de 8 dies (48 hores) i finalment s'ha tardat 12 dies (72 hores) generant un increment del 50%. Aquesta desviació es va produir per causa del disseny dels controladors de les cites. Es va voler dissenyar uns algoritmes de recuperació de cites més versàtils del que s'havia previst inicialment i va generar la inversió d'un grapat més d'hores. Concretament l'algorisme d'ordenació de cites per a una posterior recuperació especificant un nombre de dies (amb cites) a recuperar va consumir bona part del temps.

7.2 Distribució econòmica

Per dur a terme el projecte sencer es van definir dos perfils: analista i programador. Inicialment es van calcular les següents hores per a cada paper:

Perfil	Hores	€ / hora
Analista	174 hores	22 €
Programador	228 hores	18'3 €
Total	402 hores	8000'4 €

Un cop finalitzat el projecte s'ha tornat a calcular l'import total atenent les desviacions d'hores en funció de cada rol.

Perfil	Hores	€ / hora
Analista	204 hores	22 €
Programador	270 hores	18'3 €

Total	474 hores	9429 €

El projecte es va pactar amb dos preus per hora que s'han mantingut després de les desviacions. Això, en la previsió inicial, s'ajustava a 402 hores per 8000 €. Un cop finalitzat el projecte s'ha realitzat un total de 474 hores. Per tant, el cost final de l'aplicació ha sigut de 9429 €. S'ha produït una desviació del 17'86% respecte el pressupost inicial.

Capítol 8

Conclusions

8.1 Pel que fa al projecte

Realitzar aquest projecte ha suposat per mi la primera presa de contacte amb el món dels dispositius mòbils. Tenia moltes ganes de realitzar alguna aplicació que entrés a explotar les novetats de la última iteració de l'*HTML*, moltes de les quals anaven dirigides al creixent entorn dels dispositius mòbils. És per aquest motiu que la realització del projecte em va semblar molt interessant des del principi.

Les meves expectatives de com podria ser el desenvolupament eren que, tenint en compte que el món mòbil ha nascut relativament fa poc, seria una experiència diferent de com es treballa quan es realitzen aplicacions web d'escriptori. M'esperava una homogeneïtat en les noves apis del sistema i uns comportament entre dispositius molt similars. Això em feia estar emocionat per poder fer un projecte on no caldria invertir hores de proves entre navegadors per comprovar que tota la implementació que s'està fent funciona correctament en totes les plataformes. Al final, per culpa d'un sistema molt lent d'implantació d'estàndards, m'he trobat disparitat de comportaments i un cop més, per decepció meva, el projecte es va convertir en un trencaclosques de proves per assegurar una compatibilitat entre les diferents plataformes.

8.2 Pel que fa a l'aplicació resultant

En aquest apartat estic molt content del resultat obtingut. Han sigut moltes hores d'esforç per dissenyar i implementar una aplicació robusta i molt usable. La prova de la bona feina ha sigut que durant les etapes de proves i de seguiment no ha aparegut cap tipus de problema. No s'ha tingut cap queixa per part dels usuaris i prova que les hores invertides han donat bon resultat.

Per altra banda, l'ús de l'aplicació entre els comercials ha anat creixent en poc temps i sembla que finalment s'ha tornat una eina útil per als usuaris finals. Tenint en compte que aquest era l'objectiu final de l'aplicació es podria qualificar de bon resultat el treball realitzat.

8.3 Pel que fa a la tecnologia

Seguint des del punt anterior, la realitat, un cop més, la realitat tecnològica va ser decebedora. Si bé és cert que el factor de no haver de fer l'aplicació compatible amb versions d'*Internet Explorer* totalment obsoletes era un punt a favor, la poca evolució dels navegadors mòbils va fer que la implementació del sistema es tornés un galimaties a base de proves entre plataformes i navegadors per assegurar la màxima compatibilitat.

Per altra banda ha sigut interessant comprovar de primer mà que el món mòbil no es cap utopia d'homogeneïtat ni cap paradís dels estàndards. És un món al que encara li queda molt recorregut, més inclús que als navegadors d'escriptori, ja que a part de les petites disconformitats aquí i allà, s'ha de sumar el fet que els dispositius encara són lents i certes rutines o llibreries complexes encara no són viables d'utilitzar en plataformes mòbil. També he pogut comprovar de primera mà com els navegadors flaquegen de certa falta d'optimització a l'hora, per exemple, d'implementar un petit efecte d'ombra sota una caixa de text (cas real sota Safari).

Pel que fa l'API dels esdeveniments "touch" és curiós veure com, un cop més, cada navegador implementa certes diferencies respecte els altres per seguir complicar l'existència dels programadors tal com venia passant des de sempre en el món web. Els problemes amb els esdeveniments de lliscar els dits per la pantalla van ser complicats de resoldre en l'entorn d'iOS i Safari, havent de recorre a estratègies rocambolesques per esquivar els efectes indesitjats d'una API poc consistent.

Com a punt positiu es pot mencionar el fet de començar a treballar amb la nova memòria cau que permet la última versió de l'*HTML*. Ràpidament, però, veus que, un cop més, i de forma poc justificable, que una solució que es portava molt temps esperant ha sigut implementada d'una manera que força el programador, un cop més, a buscar estratègies per acabar obtenint un funcionament racional. El problema en concret és que la memòria cau només permet emmagatzemar cadenes de text. No permet emmagatzemar objectes, ni tan sols els natius. Això obliga a que quan es recupera la informació, s'ha de tractar tota per reconstruir les estructures originals.

8.4 Pel que fa al grup de treball

En aquest apartat l'experiència ha sigut molt positiva. Ha sigut molt interessant interactuar amb altres parts del departament per pactar els serveis que necessitaria per poder complir tots els requeriments que es demanaven i per pactar sistemes d'intercanviar informació.

Valoro molt positivament, també, les hores de treball conjuntes per aconseguir fer funcionar la comunicació entre el client i el servidor, realitzant proves i depurant les seqüències per trobar quines coses fallaven.

Va ser interessant el fet de que certes parts de l'aplicació s'anaven desenvolupant simultàniament amb la incorporació d'aquestes funcionalitats a la part ja existent de Seges. Això feia que a vegades el desenvolupament en cada etapa no fos seguit perquè feia falta esperar que les parts del servidor estiguessin llestes per poder implementar la comunicació d'aquell servei en concret.

8.5 Pel que fa al tracte amb el client

Jo ja havia tingut tracte amb clients en altres projectes més petits, però normalment les peticions d'aquests eren raonables i consistents amb el que es demanava dins el projecte. També és cert que certes vegades m'havia topat amb clients que demanaven coses inversemblants i s'havia d'explicar-los bé perquè no es podia o quines implicacions tenia aquella decisió.

En aquest projecte va ser interessant assistir a un parell de reunions on es va ensenyar l'estat del desenvolupament de l'aplicació i on el client, a cada reunió, anava canviant certs requeriments funcionals i anava demanant noves característiques, en certs casos, impossibles d'implementar o extremadament desviades respecte la ruta establerta en reunions anteriors. Tot això constata de nou que quan els requeriments no es deixen completament tancats des del principi s'acaben generant situacions complicades que acaben conduint al programador a treballar més del compte.

8.6 Pel que fa a la metodologia

Aquest és un punt del que realment he quedat molt satisfet. El fet de dur un seguiment constant del desenvolupament i forçant presentacions i reunions en cicles de dues setmanes m'ha ajudat molt a treballar amb uns objectius molt clars i evitant una dispersió en l'ordre d'implementació de la llista de requeriments.

També ha sigut interessant interpretar com una metodologia el fet que dues persones s'asseguin juntes davant una màquina i comencin a trobar solucions als problemes que vinculen el desenvolupament entre dos persones o equips

La metodologia Scrum o la tècnica Pair Programming han resultat ser molt eficients i els intentaré incorporar en futurs projectes.

8.7 Pel que fa a l'enriquiment personal

Finalment la realització d'aquest projecte no ha sigut l'experiència que jo esperava, per culpa de la tecnologia, que ha acabat convertint un projecte que es presentava com una feina lineal, en un conjunt de maldecaps solucionats a base de prova i error en quan a interfície es refereix. En altres capes del desenvolupament, al final ha acabat essent molt semblant a una aplicació web d'escriptori i l'única diferència l'ha marcat tot el tema de la comunicació amb el servidor i tota la lògica per controlar la dependència de cobertura del dispositiu.

Si que ha suposat un enriquiment tots els factors que envolten un projecte dins una gran empresa: la presa de requeriments, l'anàlisi, l'estudi de diferents tecnologies, les reunions de seguiment, els temps ajustats a un calendari per poder complir amb els pressupostos, les fases de proves amb els usuaris, etc. Ha sigut interessant veure tots els factors que intervenen en el desenvolupament d'una aplicació i com al final acaba sorgint un producte ben estructurat, que funciona d'una forma robusta, i que realitza el seu objectiu sense generar cap queixa per part del client.

Capítol 9

Bibliografia

[W3C] World Wide Web Consortium (W3C), http://www.w3.org/

[MSCH] DHTMLX Mobile Scheduler, http://docs.dhtmlx.com/doku.php?id=dhtmlxscheduler:mobile

[JQM] jQuery mobile, http://api.jquerymobile.com/

[W3S] W3Schools Online Web Tutorials, http://www.w3schools.com/

[STCH] Sencha Touch, http://docs.sencha.com/touch/2.2.1/

[IS4] iScroll 4, http://cubiq.org/iscroll-4

[MBDT] Mobiscroll - Date & Time Scroller, http://mobiscroll.com/component/datetime

[HTML5R] HTML5 Rocks, http://www.html5rocks.com

[CIU] Can I use..., http://caniuse.com/

[SO] Stack Overflow, http://stackoverflow.com/

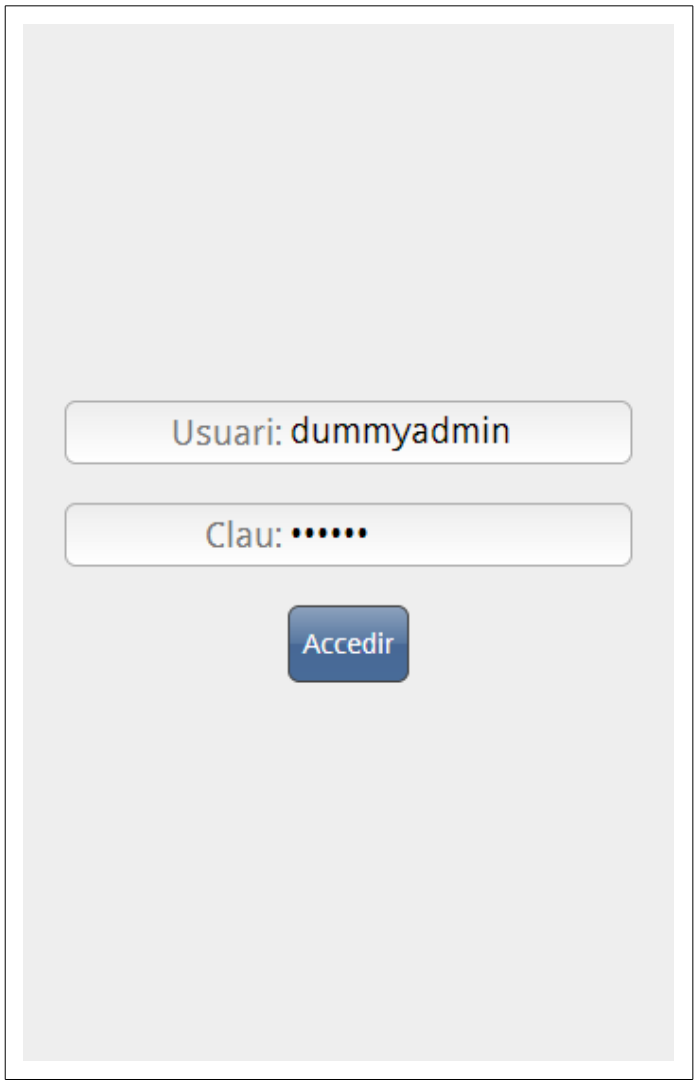
Capítol 10

Annexos

10.1 Manual de les vistes

10.1.1 Vista d'autentificació

Si és la primera vegada que s'accedeix a l'aplicació, aquesta serà la primera vista que ens



trobarem. És una vista simple amb dos requadres de text i un botó. El primer requadre correspon a l'espai on es posa el nom d'usuari. El segon requadre correspon a l'espai on s'introdueix la contrasenya. Aquest requadre està degudament configurat per mostrar uns punts enlloc dels caràcters que s'hi introdueixen.

És important notar que l'aplicació guarda el testimoni de la sessió en una galeta. D'aquesta manera la validació és permanent i només es requereix una nova validació en cas d'efectuar una desconnexió.

10.1.2 Vista de llista

Aquesta és la vista principal on s'accedeix després d'iniciar sessió o la primera vista en cas

d'haver iniciat sessió anteriorment. Ens mostra per defecte i en forma de llista totes les properes cites a partir de la data actual fins un màxim de 8 dies.

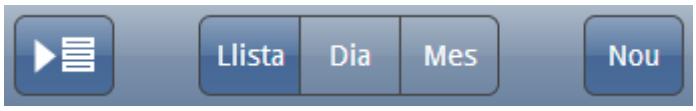
Per modificar la data a partir de la qual es volen visualitzar les cites s'ha d'accedir a alguna de les altres dues vistes principals i seleccionar un dia diferent al present. Un cop realitzada aquesta acció, al tornar a la vista de llista es visualitzaran les cites a partir del dia seleccionat.

Complementàriament, i per comoditat de l'usuari, la vista permet carregar més dies, tant en el futur com en el passat. Per efectuar tal acció tenim dues formes. Podem polsar sobre els primer o l'últim element de la llista (amb un format especial i un text bastant descriptiu) causant que es carreguin 8 elements en el futur o en el passat. També és possible desplaçar la llista més enllà dels seus límits superior o inferior. Tal acció activarà una fletxa, la qual rotarà, i d'aquesta manera indicarà que en el moment d'aixecar el punter (dit en cas de pantalla tàctil) es realitzarà l'acció de

carregar més dies. Aquesta acció es pot repetir tants cops com es vulgui. Si ja no queden més dies amb cites per mostrar, senzillament no passarà res.

La vista es divideix en dues parts:

- Zona compresa entre la part superior de la pantalla i la barra inferior: aquesta zona conté les cites distribuïdes en dies. D'entrada es mostren un màxim de 8 dies amb les seves cites (no importa la quantitat de cites per dia). Permet fer desplaçament vertical en cas de que la quantitat de cites sigui superior a la que es pot visualitzar per pantalla.
- Barra inferior amb els següents botons d'esquerra cap a dreta: *Opcions*, *Llista*, *Dia*, *Mes* i Nou.

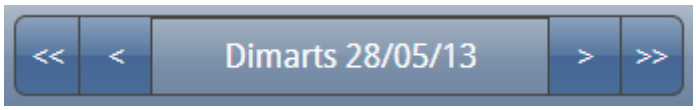


10.1.3 Vista de dia

Aquesta vista permet visualitzar totes les cites d'un dia seleccionat. Forma part de les 3 vistes principals i comparteix part de la interfície dels botons. Concretament la barra inferior.

La vista es divideix en tres parts:

- Barra superior de navegació amb els següents botons d'esquerra cap a dreta: *Mes enrere*, *Dia enrere*, *Dia endavant*, *Mes endavant*. En la part central es mostra la data actual.

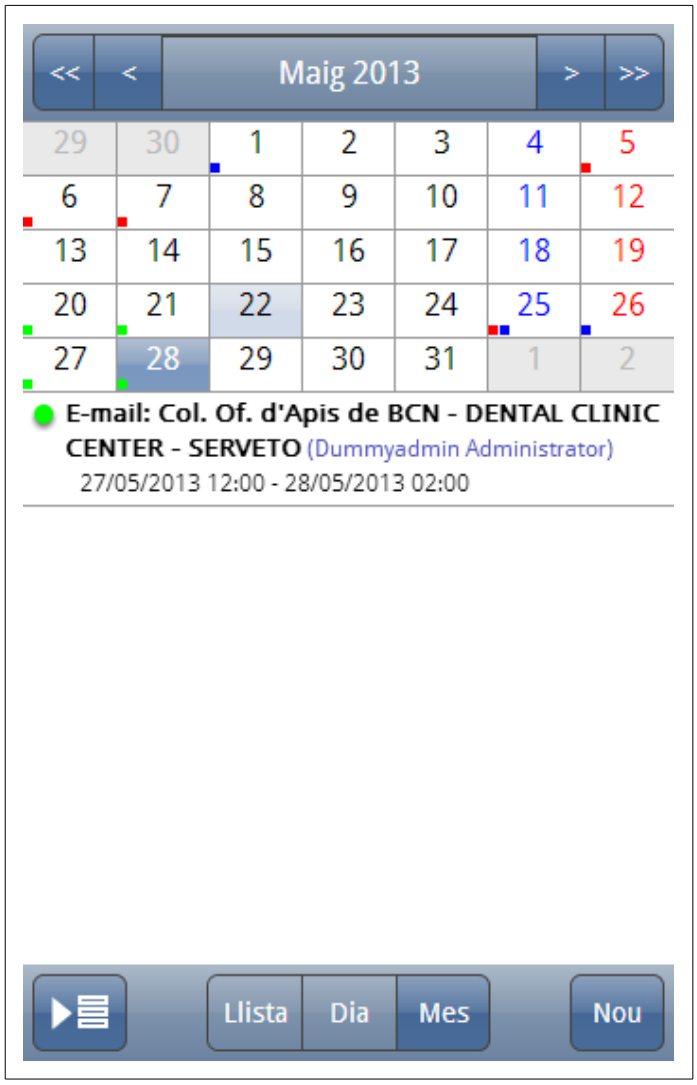


- Barra inferior de navegació amb els següents botons: *Opcions*, *Llista*, *Dia*, *Mes* i *Nou*.

- Zona central (compresa entre les dues barres). Aquesta zona conté les cites i permet fer desplaçament vertical en cas de que la quantitat de cites sigui superiors a la que es pot visualitzar per pantalla.

10.1.4 Vista mes

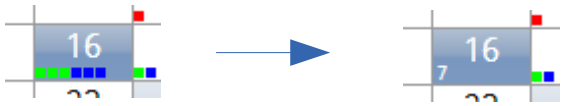
Aquesta vista ens permet visualitzar amb un cop d'ull les cites programades per a tot el més.



La vista es divideix en 4 zones:

- Barra de navegació superior amb els següents botons d'esquerra cap a dreta: *Any enrere*, *Mes enrere*, *Mes endavant*, *Any endavant*. En la part central es mostra el mes i any actuals.

- Zona del calendari: en aquest espai es visualitzen tots els dies del mes alineats per setmanes. En cas que el mes comenci o acabi a mitja setmana, la resta de la setmana serà omplerta amb dies del mes anterior o posterior.
  Els dies que no pertanyen al mes tenen un color gris pàl·lid per un major contrast visual. El dia seleccionat és d'un color blau fort, i el dia present està indicat amb un blau més pàl·lid.
  En la casella de cada dia podem veure uns punts que indiquen les cites programades per al dia en qüestió. Si el nombre de cites és superior a sis, apareixerà una marca numèrica per indicar la quantitat de cites.

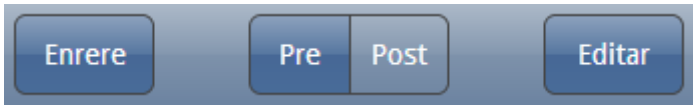


  Si es clica sobre sobre qualsevol dia, en la zona de les cites es mostren les corresponents al dia seleccionat. En cas de clicar sobre un dia pertanyent a un altra mes, la vista automàticament canvia al mes corresponent.
- Zona de les cites: espai amb desplaçament vertical on es mostren totes les cites del dia seleccionat.
- Barra inferior de navegació amb els següents botons: *Opcions*, *Llista*, *Dia*, *Mes* i *Nou*.

10.1.5 Vista detall

Aquesta vista permet visualitzar tots els detalls d'una cita. Està composta principalment per dues parts:

- Barra superior de navegació amb els següents botons: *Enrere*, *Pre*, *Post* i *Editar.*



- Zona central amb desplaçament que permet visualitzar tots els detalls d'una cita. Els camps dels que es pot veure detall són els següents: *data inici*, *data final*, *estat de la cita*, *creador*, *assignats*, *assistents*, *clients*, *contactes*, *anunciant*, *tipus de gestió*, *objectius*, *mitjans* i *observacions*.

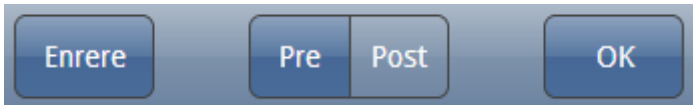

Enrere
Pre
Post
Editar
Data Inici
27/05/2013 12:00
Data Fi
28/05/2013 02:00
Estat
Tancada
Creador
Dummyadmin Administrator
Assignats
Dummyadmin Administrator
Assistents
Clients
Col. Of. d'Apis de BCN
DENTAL CLINIC CENTER
SERVETO
Contactes
Anunciant
Tipus de gestió
E-mail
Objectius
Primer contacte
Mitjans
LV
Observacions
dolor amet consectetur consectetur Lorem consectetur dolor adipiscing adipiscing sit consectetur sit consectetur sit adipiscing Lorem sit

10.1.6 Vista edició

Aquesta vista permet l'edició de tots els camps que es poden veure en la vista detall. S'hi pot accedir de dues maneres; quan triem editar una cita des del detall d'aquesta o quan creem una cita nova.

Està composta de tres parts:

- Barra superior amb els següents botons: *Enrere*, *Pre*, *Post*, *Ok*.



- Zona central amb desplaçament que permet visualitzar tots els elements editables directament o via desplaçament en cas que no apareguessin tots en pantalla.
- Barra inferior amb el botó d'eliminar.

Els camps editables són els següents:

- **Data inici** i **data fi**: quan cliquem sobre algun dels dos camps per modificar-los ens apareix la vista d'edició de temps.
- **Estat** (de la cita) i **Tipus de gestió**: quan cliquem apareix un desplegable natiu de la plataforma per poder seleccionar l'estat.
- Assignats, Assistents, Clients, Contactes, Anunciant, Objectius i Mitjans. Tots aquest camps, en ser clicats per a l'edició, condueixen a la vista d'edició de camp amb auto suggeriment.
- Observacions: és un camp de text que quan el volem editar ens porta a la vista **Edició de text**.

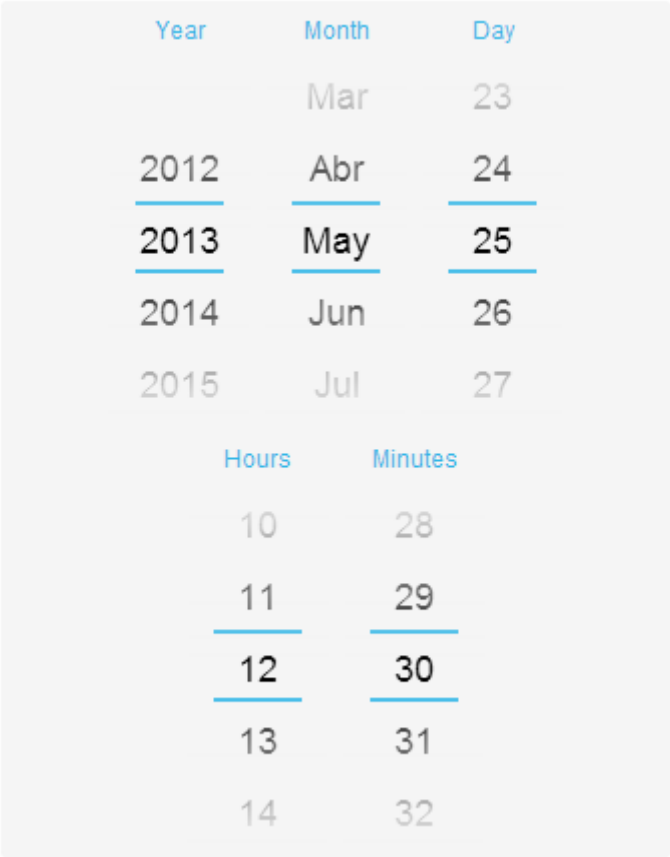
10.1.7 Vista edició de temps

En aquesta vista podem editar l'any, mes, dia, hora i minuts d'un camp temporal.

La vista es descompon en dues parts:

- Barra superior amb el botó d'enrere i Acceptar.

- Zona d'edició de la data. En aquesta zona podem trobar agrupats els modificadors d'any, mes i dia per una banda i hora i minuts per l'altra.



10.1.8 Vista edició de text

Aquesta vista ens permet disposar de tota la pantalla per a poder introduir text còmodament. Disposa de desplaçament vertical en cas que el text superi la zona visible de la pantalla.

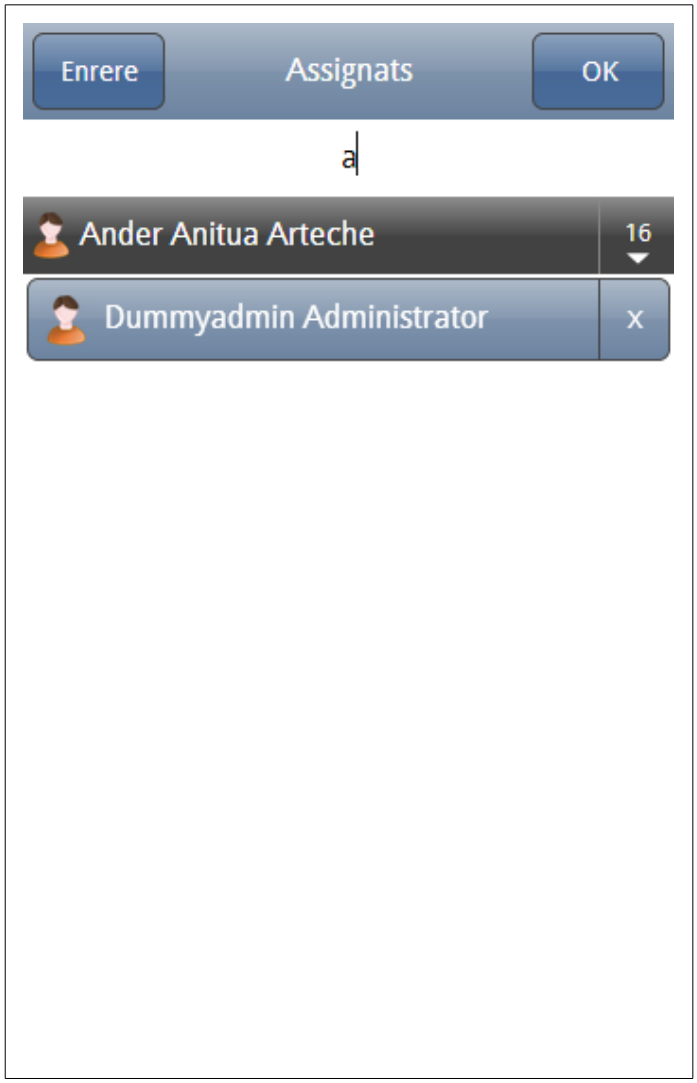
La vista es divideix en dues parts:

- Barra superior amb els botons d'enrere o acceptar (OK).

- Zona del text on podem editar el text que volem redactar. En aquesta vista existeix una barra a la part dreta de la pantalla. Aquesta barra és l'assistent per poder desplaçar la vista verticalment per poder accedir a qualsevol part del text.

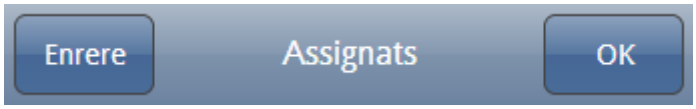
10.1.9 Vista edició camp amb auto suggeriment

En aquesta vista tenim un assistent dissenyat especialment per a les característiques mòbils. Aquest assistent ens mostra un suggeriment que compleix el text que hem introduït. Al mateix temps, ens mostra el nombre de resultats totals que coincideixen amb el text. D'aquests resultats estan exclosos els elements que ja formen part de la selecció (els que apareixen a sota). Alguns dels camps d'auto suggeriment estan vinculats a altres camps causant que la llista de resultats que es mostren estiguin filtrats des del servidor.

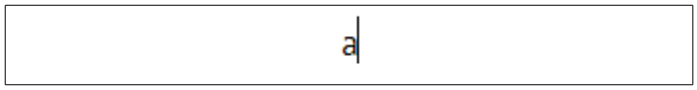


La vista es divideix en quatre zones:

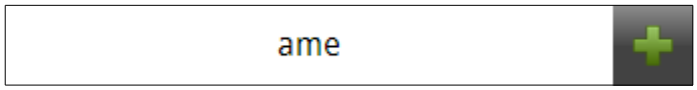
- Barra superior amb els botones d'enrere i acceptar (OK).



- Caixa de text on s'introdueix el text. Hi ha dos models: la caixa simple no permet afegir el text introduït, sinó que sempre cal seleccionar una de les opcions suggerides. La caixa ampliada mostra un símbol de més a la part dreta. Aquest símbol indica que si es polsa la tecla "introduir" en el teclat o es polsa sobre el requadre que conté el símbol de més, el text



que hi hagi dins la caixa passarà a ser un element més.



- Barra d'assistència: en la part esquerra es mostra el primer resultat que coincideix amb el

text introduït en la caixa de text. A la part dreta, en un requadre, es mostra el nombre de coincidències amb el text introduït. En aquest requadre, si apareix un nombre inferior a cinquanta, mostrarà una fletxa blanca i el requadre passarà a ser un botó polsable. Existeix

un altre estat per aquest requadre: dues fletxes en direcció oposada que indiquen que s'està produint una comunicació amb el servidor.

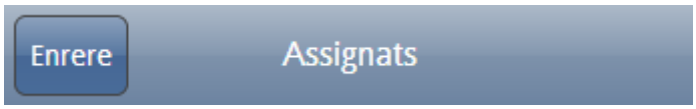
- Zona dels elements seleccionats. En aquest espai tenim una llista que permet el desplaçament vertical en cas que el nombre d'elements sobrepassi el nombre que es poden visualitzar en la pantalla. Aquesta llista està conformada pels elements que ja

formen part del camp d'auto suggeriment. Cada element mostra el nom i un botó a la part dreta per poder-lo eliminar de la llista.

10.1.10 Vista selecció en llista

Aquesta vista és un auxiliar de la vista anterior. Ens mostra una llista amb tots els resultats del suggeriment. La vista està dividida en dues parts:

- Barra superior amb el botó d'enrere i el títol del camp.



- Zona amb els elements de la llista. Aquesta llista disposa de desplaçament vertical per poder mostrar tots els elements en cas que no hi càpiguen per pantalla. Per seleccionar un element només cal polsar sobre aquest.

10.1.11 Vista opcions

Aquesta és una vista auxiliar que es va introduir per donar lloc a les diferents accions que no formaven part del funcionament habitual i per donar certa informació que tampoc tenia espai en la interfície principal. Així doncs, els elements que formen aquesta vista són els següents:

- Requadre informatiu on es mostra el nom de l'usuari que està validat i el correu.
- Netejar cau: botó que permet netejar la memòria cau de l'aplicació. Aquesta opció es va afegir per facilitar una sincronització neta descartant totes les cites emmagatzemades a la memòria local i demanant els intervals desitjats de nou.
- Sincronitzar cau: aquest botó simula l'acció de demanar al servidor totes les actualitzacions realitzades des de la última actualització. Entre parèntesis es pot veure la hora de la última actualització. L'aplicació està configurada perquè es realitzi aquesta acció de forma automàtica cada tres minuts.

- Recarregar: s'encarrega de recarregar l'aplicació. És l'equivalent de refrescar la pàgina del navegador. Tant si es fa per mitjà del navegador com si es fa polsant el botó, apareixerà un missatge alertant que es té la intenció d'abandonar la pàgina. Aquest avís ha sigut afegit

  pel costum d'utilitzar els botons d'anar endavant i enrere en l'historial del navegador. D'aquesta manera s'intenta una mica pal·liar l'efecte de tirar enrere per intentar sortir d'una cita i acabar visitant de nou la última pàgina on estàvem abans de carregar l'aplicació.
- Català i Castellà: aquests dos botons permeten canviar l'idioma de l'aplicació
- Desconnectar: serveix per eliminar tota la informació de sessió i deixar l'aplicació bloquejada en la pantalla d'inici de sessió. Si no es realitza aquesta acció de forma manual no es tancarà mai sessió ja que aquesta queda persistent en una galeta.
- Barra inferior: aquesta barra només conté el botó per sortir de les opcions.

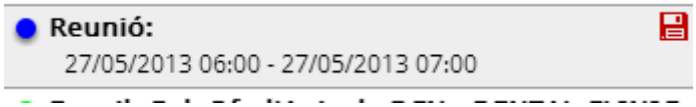
10.2 Indicadors gràfics

La interfície incorpora alguns indicadors per fer saber a l'usuari l'estat de diverses situacions del sistema:

- Indicar a l'usuari que l'aplicació **està carregant**: per a tal menester la pantalla es torna gris i mostra una roda en el centre. Això bloqueja l'aplicació durant el temps que s'està intentant comunicar amb el servidor.

- **Indicar que l'aplicació està fora de línia**: consisteix en indicar a l'usuari que l'aplicació està treballant en **mode sense connexió**. Per indicar aquesta situació, la barra inferior passa del color blau a un **gris clar**.

- **Indicar que s'ha produït un error en el procés de guardar la cita en el servidor:** aquest error pot ser causat per alguna falta de permisos o per altres motius i quedarà reflectit amb un disquet vermell a la part dreta de la cita.



- **Indicar que una cita està pendent de ser salvada en el servidor**: això quedarà reflectit amb un disquet negre a la part dreta de la cita.

10.3 Resum de funcionalitats dels controladors principals

10.3.1 CacheController

Aquest és el controlador encarregat de gestionar la interacció de l'aplicació i la memòria cau del navegador. També està encarregat de mantenir dos índexs diferents on les cites estan

ordenades seguint dos criteris diferents: indexades per identificador i indexades per dies. Dins de cada dia, les cites estan indexades per ordre de data d'inici.

Per poder dur a terme la primera tasca (emmagatzematge), i atenent les limitacions actuals en l'API *JavaScript* per a la utilització de la memòria local del navegador, s'ha dissenyat dos mètodes encarregats de guardar i recuperar aquesta informació.

- dump: comprova si el navegador té emmagatzematge local, en cas afirmatiu guarda quatre dades: data d'inici de la memòria cau, data de final de la memòria cau, última actualització de la memòria cau, i una serialització del mapa que guarda les cites mitjançant els índexs.
- restore: aquest mètode és el contrari que l'anterior i s'encarrega de recuperar la informació emmagatzemada en la memòria local del navegador.

Per poder gestionar el fet d'afegir o eliminar cites existeixen una sèrie de mètodes que faciliten la feina i s'encarreguen de mantenir els índexs al dia i perfectament sincronitzats.

- addEvent: Aquest mètode ens permet afegir una cita a tots els índexs amb una sola crida. El mètode també s'encarrega d'eliminar les cites que s'afegeixen, però que estan marcades com eliminades. Aquest funcionament s'ha implementat d'aquesta manera per un tema de delegació de tasques. Amb aquest sistema el fet de controlar si una cita s'ha d'eliminar dels índexs recau en aquest controlador evitant haver de repartir codi vinculat al format de les cites entre diferents controladors. Això permet que quan arriba un conjunt de cites des del servidor, totes són afegides sense més preocupació. Amb aquest sistema es pot indicar quan una cita ha sigut eliminada en el servidor senzillament enviant la cita amb el seu identificador i amb un camp que el controlador detecta, i que causa que la cita sigui suprimida.
- removeEvent: aquest mètode s'encarrega d'eliminar una cita de tots els índexs.

Quan una cita ha sigut modificada o eliminada s'indexa en un altre índex encarregat de guardar tots les cites modificades. Quan es fa una comunicació amb el servidor s'envien aquestes cites, i si des del servidor el dona el vist i plau, les cites són eliminades d'aquest índex. Per a gestionar aquestes accions existeixen els següents mètodes:

- saveEvent: aquest mètode rep una cita la qual, en el camp d'informació extra per a gestions internes, s'indica que té pendent una operació de guardat. Tot seguit es crida al mètode *modifiedAdd* passant-li com a paràmetre la cita. Aquesta acció afegeix la cita en un mapa indexat on es guarden totes les cites pendents de ser enviades al servidor per validar/actualitzar els canvis.
- deleteEvent: aquest mètode rep una cita com a paràmetre i la marca com que ha sigut eliminada. Després crida al mètode *modifiedAdd* passant-li com a paràmetre la cita.
- modifiedAdd: el mètode rep una cita com a paràmetre i és l'encarregat d'indexar-la en el mapa de cites modificades. La indexació es fa per identificador (evitant duplicitats).

- modifiedRemove: aquest mètode rep un identificador i s'encarrega d'eliminar del mapa de cites modificades la cita amb aquest identificador. El valor que retorna és verdader o fals en funció de si l'ha trobat indexada o no.
- modifiedGet: aquest mètode retorna el mapa que indexa totes les cites modificades.

Per últim existeix un mètode que s'encarrega de fer net la memòria cau eliminant totes les cites de tots els índexs existents.

A nivell de variables, el controlador disposa de quatre encarregades de guardar la data de les accions indicades en els noms respectius:

- cacheStart: indica en quina data comença la memòria cau. El sistema estableix que els blocs de cites dins la memòria cau van de dia en dia com a mínim. És a dir, no és possible tenir dins la memòria cau totes les cites a partir de les quatre de la tarda d'algun dia. És imperatiu del sistema que la memòria cau tingui totes les cites del dia en qüestió i la data d'inici estigui apuntant a les 0:00 hores del dia indicat.
- cacheEnd: guarda la data on finalitza la memòria cau. Segueix el mateix imperatiu que la variable anterior: no és possible disposar de la parcialitat de cites d'un dia. Sempre ha d'apuntar a les 23:59 amb 59 segons i 999 mil·lèsimes de l'últim dia de la memòria cau.
- cacheLastUpdate: guarda la data de l'última actualització de la memòria cau.
- cacheLastModification: guarda la data de l'última actualització realitzada a la memòria cau (modificació de cita, nova cita o eliminació de cita).

10.3.2 DataController

Aquest és un dels controladors més importants en la lògica de l'aplicació; s'encarrega de centralitzar totes les operacions amb les cites. Per poder realitzar totes aquestes funcions té comunicació directa amb la memòria cau (*cacheController*) i amb el servidor centrar mitjançant l'API de comunicació asíncrona del navegador (*xhr* o *ajax*).

Aquest controlador té la funció de separar la lògica de l'aplicació i el sistema de cites. D'aquesta manera, totes les operacions amb cites per part de l'aplicació es fan sobre aquest controlador, i aquest s'encarrega de redirigir-la cap al servidor i la memòria cau del sistema.

Uns dels mètodes més importants d'aquest controlador són **els tres que permeten recuperar les cites**. Gràcies a aquests s'ha pogut simplificar i agrupar la lògica per recuperar les cites:

- findByInterval: aquest mètode rep un objecte *JavaScript* que agrupa els paràmetres de la crida permetent certa flexibilitat sintàctica a l'hora d'implementar el mètode. Bàsicament aquesta funció rep una data d'inici i una data de final. El que s'espera del mètode és que

retorni totes les cites que comencen, acaben o comencen i acaben dins l'interval especificat. Existeix un paràmetre extra que permet indicar si les cites es retornen agrupades en dies o sense agrupació. Aquest controlador accedeix directament a les variables del controlador *cacheController*, d'aquesta manera pot comprovar si l'interval que es demana està contingut dins la memòria cau o no. En cas afirmatiu, utilitzant els mapes que tenen indexats les cites, es recuperen les que estan compreses en l'interval desitjat i es retornen en un nou mapa. En cas contrari es realitzarà una petició al servidor demanant totes les cites corresponents a l'interval que falta. Un cop s'obté la resposta del servidor, totes les cites són processades i guardades en la memòria cau. Llavors es torna a realitzar la comprovació de l'interval. Ara si que es complirà que la memòria cau té tot l'interval que es demana i per tant no es realitzaran més comunicacions amb el servidor per solucionar aquesta crida.

- findById: aquest mètode segueix la mateixa mecànica que l'anterior; comprova si la cita amb l'identificador especificat existeix en la memòria cau. En cas afirmatiu es retorna la cita. En cas negatiu es fa una petició al servidor demanant per la cita en qüestió.
- findSince: aquest mètode també segueix la mecànica del primer: comprova a la memòria cau i si no hi ha tota la informació, es comunica amb el servidor per completar-la. Aquesta funció té un comportament particular que ha sigut replicat a la part del servidor ja que degut al disseny funcional d'aquesta, no és possible saber d'una manera simple quantes cites s'han de demanar exactament. El que permet aquest mètode és recuperar totes les cites d'una quantitat concreta de dies que tenen cites. Per exemple, es volen totes les cites a tres dies vista (tres dies amb cites); la setmana té una cita dimarts, una divendres i una dissabte. La funció retornaria totes les cites del dimarts, del divendres i del dissabte. Si dijous també tingués una cita llavors es retornarien totes les cites del dimarts, del dijous i del divendres. El propòsit d'aquest mètode es poder, d'una forma simplificada, treballar amb la vista de llista.

El controlador incorpora també dos mètodes per salvar i per eliminar cites: *saveEvent* i *removeEvent*. Aquests dos mètodes implementen una crida als mateixos mètodes però del controlador *cacheController*.

Existeix un mètode molt important que permet la sincronització amb el servidor i que rep el nom de ***sync***. Gràcies a aquest mètode s'agrupen totes les comunicacions necessàries per mantenir l'aplicació i el servidor sincronitzats. Cada cop que s'executa la funció ***sync*** s'envia al servidor la següent informació: data inicial, data final i última actualització de la memòria cau. També s'envien totes les cites que han sigut modificades. S'entén per modificades totes les cites que han sofert alguna modificació, que han sigut creades o que han sigut eliminades. El servidor processa totes les peticions i emet una resposta que consta de la següent informació:

- Per cada cita enviada en retorna la mateixa cita tal com ha quedat guardada en el servidor. S'hi afegeix una informació extra que indica el resultat de la operació. Si el resultat d'una operació de creació o modificació és "*ok*", la cita retornada és exactament la mateixa que s'ha enviat. En cas que hagi fallat quelcom en l'operació, la cita retornada és el resultat final de l'operació i serà diferent a la cita que hem enviat.
- Ens retorna també totes les cites que han sigut modificades dins l'interval de les dates que hem enviat i que compleixin que la última modificació ha sigut posterior a la data que enviem, indicant quan va ser la última actualització amb el servidor.

La sincronització de temps entre el servidor i el client és important i en l'apartat de les implementacions s'explica com ha sigut solucionat. Malgrat la solució, cal puntualitzar que si el client té una hora per darrera la del servidor és millor que si la té per davant. En el cas que estigui per darrera pot passar que es tornin a enviar cites que ja estan dins la memòria cau no donant cap més problema que el simple fet de substituir les existents per les noves. Per contra, si la data del client està en el futur pot causar que la data que s'envia com última actualització de la memòria cau estigui en el futur, causant que el servidor no informi dels canvis més recents.

10.3.3 CoreController

És el controlador principal de l'aplicació i controla la interacció entre la resta dels controladors. Aquest disposa de dues funcions internes que s'encarreguen d'inicialitzar tota la configuració del sistema en dos passos (per temes de càrrega asíncrona). En una primera etapa inicialitza tot el tema de la sincronització i indexat de les cites realitzant comunicacions amb el servidor (de forma asíncrona). Un cop està completada aquesta etapa es passa a la configuració de les vistes.

Per configurar les vistes es crea un controlador del tipus *viewController*. En la creació d'aquest controlador se li passen una sèrie de paràmetres dels que es destaquen tres que indiquen quines funcions s'han d'executar abans de canviar una vista, en el moment de canviar la vista, i després de canviar la vista.

Un segon aspecte important de la inicialització del controlador és la configuració de les vistes. Existeix un mètode que permet vincular una cadena de text a un controlador. Mitjançant aquest sistema es vinculen identificadors de codi *HTML* amb els controladors encarregats de gestionar-lo. Per un tema d'agilitat en el disseny, l'aplicació no genera de forma processal les vistes mostrades, per tant, aquestes resideixen en l'arxiu principal de l'aplicació el qual és una pàgina estàndard *HTML*.

El tercer i últim aspecte important que inicialitza el controlador és la gestió de totes les barres que apareixen en les vistes. Encara que la lògica de les accions associades a cada barra sí

està gestionada en última instància per la vista activa, el sistema de carregar les diferents barres es controla des de aquest controlador.

10.3.4 ToolbarController

Aquest és el controlador encarregat de la lògica de les barres superiors i inferiors de les diferents vistes. S'encarrega de controlar els esdeveniments que en elles s'hi produeixen de forma que es pot controlar quines acciones es poden fer i quines no. D'aquesta mateixa manera, és possible controlar agrupacions de botons de forma que només un element del grup estigui actiu i que al activar un altra, l'actiu passi a estar no actiu.

Un altre aspecte interessant és que les barres poden tenir un **text que es visualitza en la part central** a mode de títol si la quantitat de botons ho permeten.

També es disposa d'un mètode "*on*" i "*off*" per **activar** o **desactivar** la recepció d'esdeveniments. Això ha sigut implementat per evitar problemes quan la barra no està visible o té altres elements de la interfície que la solapen.

10.3.5 ViewController

La funcionalitat d'aquest controlador es pot definir com la de gestor de vistes. Bàsicament s'encarrega de mantenir un registre de totes les vistes existents i les activa i desactiva en funció del que la lògica del sistema demana.

El mètode principal del controlador és el d'afegir vistes, el qual s'utilitza en la inicialització de l'aplicació dins del controlador c*oreController*. L'acompanya el mètode per activar la vista.

El controlador també fa la funció de repetidor. Si es desitja implementar alguna lògica que requereix passar informació a totes les vistes, aquest serà el controlador encarregat de fer-ho, ja que és l'únic que coneix totes les vistes existents. La forma d'executar aquesta replicació és iterar totes les vistes i comprovar si tenen el mètode que es desitja. En cas afirmatiu, s'executa aquest reenviant els paràmetres. En cas contrari, s'ignora la vista i es segueix el procés. Els mètodes implementats que compleixen aquesta sistema han sigut els següents:

- setSelected: permet configurar la data seleccionada. Aquest mètode s'executa quan en alguna vista es selecciona una data, i permet replicar l'acció en la resta de vistes perquè quedin ben configurades.
- CacheChanged: quan s'ha produït algun canvi en la memòria cau de les cites, aquest mètode és invocat per avisar totes les vistes que tal fet s'ha produït. Cada vista decideix que fer al respecte.

- SetCommunication: permet transmetre a totes les vistes l'estat en el que es troba actualment la comunicació amb el servidor. Cada vista decideix com actuar al respecte.

10.3.6 ViewBaseController

Aquest és el controlador base per a totes les vistes. Implementa una sèrie de mètodes comuns que són "heretats" per les vistes fill. Cada vista pot reimplementar el mètode en cas que l'acció estipulada per defecte no sigui suficient. *JavaScript* no permet el sistema clàssic d'herència, i per tant no existeixen conceptes com cridar la funció del pare. Així doncs, si es vol afegir codi en algun dels mètodes i mantenir les funcions originals, s'ha de replicar tot el codi original i afegir el nou.

Els mètodes que han de compartir totes les vistes són els següents:

- on: aquest mètode s'executa quan la vista és activada. Per defecte el mètode activa una marca interna per indicar que la vista està activa. En la majoria de vistes aquesta funció està redefinida de forma que s'inicialitzin o es refresquin elements particulars de cada vista.
- off: aquest mètode s'executa quan la vista és desactivada. Per defecte desactiva la marca de vista activada. En la majoria de vistes aquesta funció també està redefinida per tractar els casos particulars.
- refresh: per defecte el mètode no fa res, però en algunes vistes està redefinit per permetre refrescar la lògica de vista en cas que des de la lògica principal es desitgi fer-ho.
- cacheChanged: aquest mètode implementa internament una crida al mètode anterior. L'objectiu d'aquesta funció és permetre comunicar a una vista que la memòria cau de les cites ha sigut modificada. En la majoria de les vistes això permet que el controlador de la vista activa actualitzi les dades que s'estan visualitzant.
- setCommunication: el mètode per defecte està buit, però permet que cada vista implementi la crida com millor li convingui. L'objectiu d'aquesta funció és permetre que la vista activa sigui informada de l'estat actual de la comunicació amb el servidor.
- bind: aquest mètode està directament heretat per tots els controladors i serveix per vincular esdeveniments a cada vista. Aprofitant l'ús de la llibreria *jQuery*, s'ha decidit utilitzar el gestor d'esdeveniments que incorpora enlloc de programar-ne un de nou. Així doncs, dins aquest controlador base, existeix un objecte *jQuery* al que s'associen els esdeveniments i sobre el qual es poden cridar els mètodes per emetre esdeveniments personalitzats o els ja existents. Per associar un esdeveniment es crida la funció "on" sobre l'objecte *jQuery* passant-li com a paràmetres els mateixos que es reben en la crida "*bind*".

- unbind: aquest mètode funciona exactament igual que l'anterior però executant la crida "off" sobre l'objecte *jQuery* contenidor dels nostres esdeveniments.
- setBackView: l'objectiu d'aquest mètode és poder indicar a un controlador de vista a quina vista s'ha de tornar quan l'esdeveniment "enrere" succeeix.
- setTitle: aquest mètode permet especificar un títol per a una vista. Aquest títol es mostra en la barra superior en cas que existeixi. No hi ha cap situació en la que s'hagi d'especificar un títol per a una vista que no té barra superior.

10.3.7 ViewOptionsController

Aquest és el controlador encarregat de gestionar la lògica de la vista "opcions". Principalment aquesta vista permet efectuar operacions de "manteniment" sobre l'aplicació i canviar l'idioma.

Existeix un mètode "*init*" que s'encarrega de configurar la vista en el moment en que es crea una vista. Seria l'equivalent del constructor en un llenguatge de programació com el c++. Aquesta funció dins aquest controlador s'encarrega d'inicialitzar els següents elements:

- L'artefacte encarregat del desplaçament vertical.
- Associar un oient al controlador principal de les cites de manera que quan hi hagi una sincronització amb el servidor, el controlador rebi una notificació i pugui actualitzar la informació de en quin moment s'ha realitzat la última sincronització amb el servidor.
- Mitjançant la llibreria *jQuery* s'associa l'esdeveniment "*clic*" de tots els botons de la vista amb la funció "*navigate*".
- Inicialitzar la barra d'eines inferior i associar l'esdeveniment de *clic* del ratolí amb la funció "*navigate*".

El mètode principal dins aquesta vista es diu "*navigate*" i s'encarrega de gestionar els esdeveniments de la vista. Quan es fa *clic* sobre la barra inferior o sobre algun dels botons que composen la vista, es fa una crida a aquesta funció, la qual, depenent de l'identificador de l'element sobre el que s'ha realitzat l'esdeveniment, tria quina acció es produeix.

Els mètodes "on" i "off" en aquest controlador han sigut reimplementats per incloure més funcionalitats de les que venen per defecte. En el mètode "on" s'hi ha afegit una crida que actualitza la hora que indica el moment de la última sincronització amb el servidor de forma que sempre quan es mostra la vista aquesta informació estigui actualitzada. En els dos mètodes també s'hi ha afegit les crides respectives per activar i desactivar la detecció d'esdeveniments en la barra (inferior).

10.3.8 ViewListController

Aquest controlador s'encarrega de gestionar la lògica dins la vista de llista. Aquesta vista permet visualitzar les cites com si d'una llista es tractés. Mostra les cites agrupades per dies i ordenades temporalment.

Una de les característiques d'aquesta vista és que permet carregar més dies (amb les seves corresponents vistes) realitzant un simple desplaçament vertical de la llista. Els mètodes que permeten realitzar aquestes accions són els següents:

- pullUpAction: aquest mètode és executat quan es detecta que la llista ha sigut arrossegada cap a la part superior de la pantalla. Això indica que l'usuari vol veure més cites per la part inferior de la llista i que per tant s'ha de carregar més dies en el futur.
- PullDownAction: aquest mètode realitza el mateix esquema que l'anterior però amb la diferència que carrega més dies en el passat.

La vista està configurada per carregar, sempre que es demanen dies en el futur o en el passat, la quantia de vuit dies. És a dir, si l'usuari arrossega la llista cap a la part superior indicant que vol carregar / veure més dies en el futur, aquesta acció carregarà els propers vuit dies que tinguin cites. En cas que no hi hagi suficients dies (perquè no existeixen més cites en el futur) es mostraran tots els dies que s'hagin pogut carregar.

Ambdós mètodes utilitzen la funció *findSince*. Gràcies a la utilització d'aquest mètode és possible carregar d'una forma senzilla un valor concret de dies.

10.3.9 ViewDayController

Aquest controlador s'encarrega de gestionar la lògica de la vista de dia. Les cites es mostren en un format de llista i permet el desplaçament vertical per si el conjunt de les cites ocupés més del que es pot visualitzar en la zona visible. Aquesta és una vista molt simple que fa ús del mètode *findByInterval* passant com a paràmetres l'inici i la fi del dia. Això recupera totes les cites presents en el dia que es mostra.

10.3.10 ViewMonthController

Aquest és el controlador de la vista de mes. S'encarrega de controlar les barres, el calendari i la llista amb cites que es poden veure en aquesta cita. Aquesta vista utilitza el controlador *slenderCalendar* per presentar el calendari interactiu.

En la funció encarregada de controlar els esdeveniments de l'usuari s'ha vinculat els esdeveniments de la barra amb els mètodes del calendari per poder avançar i retrocedir un mes o un any.

En la inicialització del controlador es configura el calendari per recollir els esdeveniments de seleccionar un dia i canvi de mes (el calendari canvia de mes de forma automàtica quan es selecciona un dels dies visibles pertanyents al mes anterior o posterior). Aquesta configuració permet a la vista saber l'estat del calendari i actuar en conseqüència; si es canvia de dia seleccionat s'ha de carregar en la llista les cites del dia escollit. Si es canvia de mes, a part de carregar les cites del dia escollit, també és necessari carregar i configurar el calendari amb totes les cites del nou més.

10.3.11 ViewDetailsController

Aquest controlador permet gestionar la vista que mostra tots els detalls d'una cita. Els mètodes més interessants que implementa per poder realitzar la seva lògica són els següents:

- retrieveEventInfo: aquest mètode es crida cada cop que s'activa la vista i permet recuperar tota la informació pertinent a la cita que es vol visualitzar.
- fillEventInfo: aquest mètode s'encarrega d'omplir els elements visuals amb la informació de la cita. Per poder omplir d'una forma còmoda els camps que són un vector de dades s'ha implementat el mètode que es descriu a continuació.
- fillEventInfoArray: aquest mètode s'implementa per poder-li delegar la funció de pintar per pantalla tots els elements d'un vector dins la vista de detalls.
- ClearEvenFields: aquest mètode és una utilitat interna per netejar la vista de qualsevol dada. S'utilitza en el moment de desactivar aquesta vista deixant tots els camps nets i preparats per quan es torni a activar.
- setId: aquest mètode permet indicar l'identificador de la cita que s'ha de mostrar quan s'activa la vista.

En el moment d'activar la vista i abans de carregar la informació de la cita es comprova si la cita és futura o ja s'ha iniciat (o ja ha acabat). Amb aquesta comprovació s'inicialitza la propietat "*mode*" que permet indicar si en els detalls de la vista s'han de mostrar els detalls el grup "*pre*" o "*post*".

Quan s'ha de carregar la informació de cita, es mira el "mode". En funció d'aquest es mostren les dades emmagatzemades en el grup "*pre*" o "*post*".

La barra superior presenta dos botons que permeten alternar entre el mode "pre" i "post". Quan la vista s'activa i després de d'iniciar la variable "mode" amb el valor que correspon, es configura la barra superior perquè es mostri actiu el botó corresponent al mode configurat.

Quan l'usuari canvia el mode de la vista, internament es crida la funció *clearEventFields* i després la funció *fillEventInfo* per tornar a omplir els camps amb el nou mode.

10.3.12 ViewEditController

Aquest controlador gestiona tota la lògica de la vista que permet editar una cita. Implementa els mètodes principals següents per poder realitzar les operacions necessàries:

- save: aquest mètode prepara una crida al mateix mètode de la classe *dataController*. En els paràmetres es configura l'acció que s'ha de realitzar en cas que la cita s'hagi guardat correctament o en cas que s'hagi produït un error.
- deleteEvent: aquest mètode prepara una crida al mateix mètode de la classe dataController.

De la mateixa manera que el controlador anterior, aquesta vista també presenta uns mètodes per poder pintar la informació que s'ha de mostrar per pantalla:

- retrieveEventInfo: a diferència del controlador anterior, aquest implementa una funcionalitat extra que consisteix en duplicar la informació recuperada. La informació es modificarà sobre la copia realitzada. En cas que es guardin tots els canvis la copia substituirà l'original.
- fillEventInfo
- fillEventInfoArray
- clearEvenFields
- setId

Aquest controlador presenta un mètode init molt extens perquè s'hi realitzen totes les configuracions que permeten que els camps siguin editables.

10.3.13 SlenderCalendar

Aquesta controlador serveix per mostrar un calendari interactiu. Aquest calendari permet detallar el número de cites que té cada dia. També permet seleccionar de forma directa el dia del qual es volen veure les cites. Els mètodes principals que implementa són els següents:

- prev, next, prevYear, nextYear: aquest mètodes permeten avançar o retrocedir el calendar un mes o un any sencer.

- date: permet indicar al calendari una data. La única funció d'aquesta data és permetre al calendari saber quin mes ha de mostrar. A efectes pràctics és indiferent passar l'1 de gener o el 27 de gener. Qualsevol de les dues dates causaria que el calendari mostri el mes de gener.
- currentDate: aquest mètode permet especificar o recuperar el dia present.
- selectedDate: aquest mètode permet especificar o recuperar el dia seleccionat en el calendari.
- setMarks: aquest mètode permet configurar un conjunt de marques que es dibuixaran en el calendari. Aquest mètode rep un mapa indexat amb el sistema d'indexació de dies explicat en anteriors punts. Cada element del mapa és un vector que guarda informació per cada marca que s'ha de dibuixar en aquell dia.

En el punt 10.1.4 de l'annex de la descripció de les vistes es pot consultar més informació sobre l'aspecte visual.